(19) BUNDESREPUBLIK DEUTSCHLAND

DEUTSCHES PATENT- UND MARKENAMT

(12) **Veröffentlichung**

(10) **DE 100 85 324 T 1**

(51) Int. Cl.[7]: G 06 F 11/16

der internationalen Anmeldung mit der
(87) Veröffentlichungsnummer: WO 01/46806 in deutscher Übersetzung (Art. III § 8 Abs. 2 IntPatÜG)
(21) Deutsches Aktenzeichen: 100 85 324.2
(86) PCT-Aktenzeichen: PCT/US00/41079
(86) PCT-Anmeldetag: 4. 10. 2000
(87) PCT-Veröffentlichungstag: 28. 6. 2001
(43) Veröffentlichungstag der PCT-Anmeldung in deutscher Übersetzung: 5. 12. 2002

(30) Unionspriorität:
09/469,963 21. 12. 1999 US

(71) Anmelder:
Intel Corporation, Santa Clara, Calif., US

(74) Vertreter:
Zenz, Helber, Hosbach & Partner, 45128 Essen

(72) Erfinder:
Quach, Nhon Toai, San Jose, Calif., US

Prüfungsantrag gem. § 44 PatG ist gestellt

(54) Firmwaremechanismus zum korrigieren von weichen Fehlern

DE 100 85 324 T 1

DE 100 85 324 T 1

Die folgenden Angaben sind den vom Anmelder eingereichten Unterlagen entnommen

PATENTANWÄLTE ZENZ, HELBER, HOSBACH & PARTNER · HUYSSENALLEE 58-64 · D-45128 ESSEN

1768
8-bo/ip

PCT/US00/41079

FIRMWAREMECHANISMUS ZUM KORRIGIEREN VON WEICHEN FEHLERN

Hintergrund der Erfindung

Technisches Gebiet Die vorliegende Erfindung betrifft Mikroprozessoren und insbesondere Mikroprozessoren, die in hochgradig zuverlässigen Betriebsmoden arbeiten können.

Hintergrundinformationen Weiche Fehler treten auf, wenn Alphateilchen oder kosmische Strahlungen auf eine integrierte Schaltung treffen und an den Spannungsknoten der Schaltung gespeicherte Ladungen ändern. Wenn die Ladungsänderung groß genug ist, kann eine einen logischen Zustand repräsentierende Spannung in eine einen anderen logischen repräsentierende Spannung geändert werden. Beispielsweise kann eine einen logischen Wahr-Zustand repräsentierende Spannung in eine einen logischen Falsch-Zustand repräsentierende Spannung geändert werden, und alle den logischen Zustand enthaltenden Daten werden verfälscht.

Die Fehlerrate für weiche Fehler (soft error rate-SER) in integrierten Schaltungen wie Mikroprozessoren ("Prozessoren") nimmt mit der Entwicklung der Halbleiterprozeßtechnologien hin zu kleineren Abmessungen und niedrigeren Betriebsspannungen zu. Kleinere Prozeßabmessungen ermöglichen, daß höhere Bauelementdichten auf dem Prozessorchip erzielt werden können. Dies erhöht die Wahrscheinlichkeit dafür, daß ein Alphateilchen oder kosmische Strahlung auf einen der Spannungsknoten des Prozessors auftrifft. Niedrigere Betriebsspannungen bedeuten, daß Durchschläge von geringeren Ladungen genügen, um den von den Knotenspannungen repräsentierten logischen Zustand zu ändern. Beide Entwicklungen weisen auf höhere SERs in der Zukunft hin. Weiche Fehler können in einem Prozessor korrigiert werden, wenn sie erfaßt

werden, bevor verfälschte Ergebnisse zur Aktualisierung des architektonischen Zustands des Prozessors verwendet werden.

Prozessoren verwenden häufig paritätsabhängige Mechanismen, um eine Datenverfälschung aufgrund von weichen Fehlern zu erfassen. Jedem Datenblock wird bei der Speicherung ein Paritätsbit zugeordnet. Das Bit wird in Abhängigkeit davon auf Eins oder Null gesetzt, ob es in dem Datenblock eine ungrade oder gerade Anzahl von Einsen gibt. Wenn der Datenblock von seinem Speicherplatz ausgelesen wird, wird die Anzahl der Einsen in dem Block mit dem Paritätsbit verglichen. Eine Diskrepanz zwischen den Werten zeigt an, daß der Datenblock verfälscht wurde. Die Übereinstimmung der Werte zeigt an, daß entweder keine Verfälschung aufgetreten ist oder zwei (oder vier ...) Bits geändert wurden. Da die zuletzt genannten Fälle sehr geringe Wahrscheinlichkeiten haben, liefert die Parität einen zuverlässigen Hinweis dafür, ob Daten verfälscht wurden. Fehlerkorrigierende Codes (Error correcting codes-ECCs) sind paritätsabhängige Mechanismen, welche zusätzliche Informationen für jeden Datenblock verfolgen. Die zusätzlichen Informationen ermöglichen das Identifizieren und Korrigieren des (der) verfälschten Bit(s).

Parität/ECC-Mechanismen wurden bei Cache-Speichern, Speichern und ähnlichen Datenspeicheranordnungen umfangreich eingesetzt. Diese Strukturen haben relativ hohe Datenspeicherknotendichten und sind bereits bei den derzeitigen Abmessungen der Bauelemente anfällig für weiche Fehler. Ihre lokalisierten Arraystrukturen machen es relativ einfach, Parität/ECC-Mechanismen zu implementieren. Zu den weiteren Schaltungen auf einem Prozessor gehören Datenpfade, eine Steuerlogik, eine Ausführungslogik und Register ("Ausführungskern bzw. execution core"). Die unterschiedlichen Strukturen dieser Schaltungen und ihre Verteilung über den Prozessorchip machen es schwieriger, Parität/ECC-Mechanismen anzuwenden.

Eine Möglichkeit zum Erfassen von weichen Fehlern in einem Ausführungskern besteht darin, Befehle auf doppelten Ausführungskernen zu verarbeiten und die von beiden bestimmten Ergebnisse Befehl für Befehl zu vergleichen ("redundante Ausführung"). Beispielsweise enthält ein Computersystem zwei separate Prozessoren, die derart gebootet werden können, daß sie entweder in einem symmetrischen Mehrprozessorbetriebs("symmetric multi-processing-SMP")-Betriebsmodus oder einem funktional redundanten Prüf("Functional redundant Check-FRC")-Betriebsmodus arbeiten können. Im SMP-Modus wird die Befehlsausführung auf die Prozessoren verteilt, um eine höhere Gesamtleistungsfähigkeit gegenüber Einprozessorsystemen zu erzielen. Im FRC-Modus führt ein Prozessor einen Code normal aus, und ein zweiter Prozessor führt identische Befehle an den gleichen Daten aus, die dem ersten Prozessor geliefert werden. Wenn der zweite Prozessor eine Diskrepanz zwischen seinen Operationen und denjenigen des ersten Prozessors erfaßt, meldet er einen Fehler. Der Betriebsmodus kann nur durch Rücksetzen (resetting) des Computersystems zwischen dem SMP-Modus und dem FRC-Modus umgeschaltet werden.

Die Zweiprozessorvariante ist kostspielig (im Hinblick auf das Silizium). Darüber hinaus ist die zum Vergleich der Ergebnisse ausgeführte Signalgebung zwischen den Prozessoren zu langsam, um verfälschte Daten zu erfassen, bevor sie architektonische Zustände des Prozessors aktualisieren. Infolge dessen ist diese Variante nicht geeignet, um die erfaßten weichen Fehler zu korrigieren.

Ein weiteres Computersystem ermöglicht eine Ausführungsredundanz, indem zwei Ausführungskerne auf einem Einprozessorchip verwendet werden. Die beiden Ausführungskerne arbeiten im FRC-Modus, und ECC-geschützte Fixpunkt- bzw. Checkpointregister speichern Informationen über Zwischenzustände des Prozessors. Wenn ein Fehler in einem Codesegment erfaßt wird, führt der Prozessor eine Mikrocoderoutine aus, um mit

Hilfe der Checkpointregister den letzten unverfälschten Prozessorzustand für den Prozessor wiederherzustellen. Die Steuerung wird dann an das Codesegment zurückgegeben, das mit dem (den) Befehl(en) beginnt, der (die) auf den Fehler gestoßen war(en). Die Fehlerbehebungsmikrocoderoutine ist auf dem Prozessorchip gespeichert, wodurch deren Aktualisierung oder Modifizierung schwierig ist. Außerdem sind Routinen, die flexibel genug sind, um ein breites Spektrum von Fehlern zu korrigieren in der Regel relativ komplex. Mikrocodeimplementierungen dieser Routinen belegen einen beträchtlichen Bereich auf dem Prozessorchip.

Die vorliegende Erfindung wendet sich diesen und anderen Mängeln von erhältlichen hochgradig zuverlässigen Computersystemen zu.

Zusammenfassende Darstellung der Erfindung

Die vorliegende Erfindung betrifft einen Firmwaremechanismus, um weiche Fehler in einem Prozessor mit zwei Ausführungskernen (dual execution core processor) zu beheben, welcher die Ausführungskerne im redundanten Modus (redundant mode) und im geteilten Modus (split mode) arbeiten lassen kann.

Ein erfindungsgemäßes Verfahren erfaßt einen weichen Fehler, wenn der Prozessor die Ausführungskerne im redundanten Modus arbeiten läßt. Auf jedem Ausführungskern wird eine Fehlerbehebungsroutine ausgeführt, um unverfälschte Daten aus dem Ausführungskern zugeordneten Speicherstrukturen zu sichern. Mit Hilfe der gesicherten Daten werden die Prozessorzustandsdaten wiederhergestellt, und der erste und zweite Ausführungskern werden mit Hilfe der Prozessorzustandsdaten initialisiert.

Ein erfindungsgemäßes Computersystem enthält einen Prozessor mit zwei Ausführungskernen und einen die Fehlerbehebungsroutine speichernden nicht flüchtigen Speicher. Die Fehlerbehebungsroutine wird aufgerufen, wenn der Prozessor

einen weichen Fehler erfaßt, während er in dem redundanten Modus arbeitet. Die Routine schaltet den Prozessor auf den geteilten Modus um, wobei jeder Ausführungskern in diesem Modus unverfälschte Daten in seinen zugeordneten Speicherstrukturen an einem bestimmten Speicherplatz sichert. Die Routine stellt die Prozessorzustandsdaten auf der Basis der gesicherten Daten wieder her.

Kurzbeschreibung der Zeichnungen

Die vorliegende Erfindung wird unter Bezugnahme auf die folgenden Zeichnungen verständlich, in denen auf gleiche Elemente mit gleichen Zahlen hingewiesen wird. Diese Zeichnungen sollen ausgewählte Ausführungsbeispiele der vorliegenden Erfindung veranschaulichen und dienen nicht zur Einschränkung des Schutzbereichs.

Figur 1 zeigt ein Blockschaltbild eines Ausführungsbeispiels eines für die Implementierung der vorliegenden Erfindung geeigneten Computersystems.

Figur 2A zeigt ein Blockschaltbild eines Ausführungsbeispiels des Prozessors mit zwei Ausführungskernen gemäß Figur 1.

Figur 2B zeigt ein Blockschaltbild eines Ausführungsbeispiels der FET-Stufe des Prozessors gemäß Figur 2A.

Figur 2C zeigt ein Blockschaltbild eines Ausführungsbeispiels der Prüfeinheit des Prozessors gemäß Figur 2A.

Figur 3 zeigt ein Ablaufdiagramm, das ein Ausführungsbeispiel eines erfindungsgemäßen Verfahrens zum Beheben von weichen Fehlern darstellt.

Figur 4 zeigt ein Ablaufdiagramm, das ein Ausführungsbeispiel einer von einem Ausführungskern implementierten Fehlerbehebungsroutine darstellt.

Figuren 5A und 5B zeigen Blockschaltbilder, welche Ausführungsbeispiele für verschiedene Mechanismen zum Abstimmen (reconciling) gesicherter Daten zur Behebung von weichen Fehlern darstellen.

Figur 6 zeigt ein Ablaufdiagramm, das ein Ausführungsbeispiel des Datenabstimmungsmechanismusses gemäß Figur 5A darstellt.

Figur 7 zeigt ein Blockschaltbild eines Ausführungsbeispiels eines Ausführungskerns, welcher eine Clusterebenenredundanz ermöglicht, wenn ein Prozessor mit zwei Kernen im geteilten Modus betrieben wird.

Detaillierte Beschreibung der Zeichnungen

In der folgenden Beschreibung sind zahlreiche spezielle Details angegeben, um ein vollständiges Verständnis der Erfindung zu ermöglichen. Für den Durchschnittsfachmann, dem diese Beschreibung zugute kommt, ist jedoch klar, daß die Erfindung ohne diese speziellen Details ausgeführt werden kann. Ferner wurden verschiedene bekannte Verfahren, Prozeduren, Komponenten und Schaltungen nicht im Detail beschrieben, um die Aufmerksamkeit auf die Merkmale der vorliegenden Erfindung zu richten.

Die vorliegende Erfindung schafft eine flexible Möglichkeit, um weiche Fehler in einem Prozessor mit zwei Kernen zu korrigieren, welcher dynamisch zwischen einem redundanten Modus und einem geteilten Modus hin und her geschaltet werden kann. Im redundanten Modus läßt der Prozessor die Ausführungskerne im Lock-Step bzw. Verriegelungsschritt an identischen Codesegmenten arbeiten und vergleicht die Ergebnisse, um Fehler zu erkennen. Im geteilten Modus können die Ausführungskerne Befehle unabhängig voneinander verarbeiten, z. B. können die Ausführungskerne verschiedene Befehle in einem vorgegebenen Taktzyklus bearbeiten.

Bei einem erfindungsgemäßen Ausführungsbeispiel kann der redundante Modus ein hochgradig zuverlässiger (high reliability-HR) Prozessorausführungsmodus sein, der die Gefahr von weichen Fehlern bei der Ausführung von kritischen Codesegmenten verringert. Der geteilte Modus kann ein Hochleistungs(high performance-HP)-Prozessorausführungsmodus sein,

der verfügbar ist, um ausgewählten Code schneller zu verarbeiten, indem die dem ausgewählten Code verfügbaren Ausführungsressourcen erhöht werden. Der geteilte Modus kann ferner nur für ausgewählte Zwecke verfügbar sein, beispielsweise für die Fehlerbehebung oder für das Bootstrapping des Prozessors. Der wesentliche Aspekt des geteilten Modus besteht darin, daß es dieser den Ausführungskernen ermöglicht, "unabhängig" zu arbeiten, d. h. jeder kann verschiedene Befehle in einem vorgegebenen Taktzyklus ausführen.

Gemäß der vorliegenden Erfindung wird eine Fehlerbehebungsroutine in einem nicht flüchtigen Speicher gespeichert. Auf die Fehlerbehebungsroutine wird zugegriffen, wenn der eine Programmteilprozeß bzw. einen Programm-Thread im redundanten Modus ausführende Prozessor einen weichen Fehler erfaßt. Der Fehler kann durch eine Diskrepanz zwischen den Ergebnissen in den beiden Ausführungskernen angezeigt werden. Die Routine schaltet den Prozessor auf den geteilten Modus um, in welchem Modus jeder Ausführungskern seine zugehörigen Speicherplätze auf verfälschte Daten hin untersucht. Unverfälschte Daten werden an einen bestimmten Speicherplatz kopiert, und es werden genügend Prozessorzustandsdaten aus den unverfälschten Daten wiederhergestellt, um den unterbrochenen Programmteilprozeß wiederaufzunehmen. Die Fehlerbehebungsroutine bzw. Wiederherstellungsroutine initialisiert die Ausführungskerne mit den wiederhergestellten Prozessorzustandsdaten.

Bei einem erfindungsgemäßen Ausführungsbeispiel sind verschiedene jedem Ausführungskern zugeordnete Ressourcen paritätsgeschützt, und die Wiederherstellungsroutine implementiert eine Paritätsprüfung, um verfälschte Daten in einem Ausführungskern zu erkennen. Zu den paritätsgeschützten Ressourcen können beispielsweise Universalregister, Gleitkommaregister, Steuer- und Statusregister, Cache-Speicher niedriger Ebene (low level caches) und der gleichen gehören. Im allgemeinen kann jede zum Speichern von Prozessorzustandsda-

ten verwendete Struktur paritätsgeschützt sein. Der Umfang des Schutzes hängt von den Zuverlässigkeitsanforderungen des Systems ab.

Bei einem weiteren erfindungsgemäßen Ausführungsbeispiel erkennt die Fehlerbehebungsroutine unverfälschte Daten aus einem oder aus beiden Ausführungskernen und sichert diese an einem bestimmten Speicherplatz. Die Ausführungskerne werden mit aus den gesicherten Daten wiedergewonnenen Prozessorzustandsdaten initialisiert. Beispielsweise können beide Ausführungskerne mit den Prozessorzustandsdaten von dem Ausführungskern initialisiert werden, welcher nicht von dem weichen Fehler betroffen war. Alternativ können unverfälschte Daten von beiden Ausführungskernen an verschiedenen Speicherplätzen gesichert werden, und Daten von einem Speicherplatz können verwendet werden, um die verfälschten Daten an dem anderen Speicherplatz zu ersetzen. Bei dieser Alternative wird jeder Ausführungskern mit einer Kopie der Prozessorzustandsdaten aus seinem bestimmten Speicherplatz initialisiert. Diese Alternative verringert die Gefahr, daß ein während des Initialisierungsprozesses erzeugter weicher Fehler nicht erkannt wird, indem separate Sätze von Prozessorzustandsdaten geführt werden. Die Wiedergewinnungsroutine synchronisiert die Ausführungskerne für die Rückkehr zum redundanten Modus nach der Erkennung und der Abstimmung von gesicherten Prozessorzustandsdaten aus den Ausführungskernen im geteilten Modus.

Bei einem weiteren erfindungsgemäßen Ausführungsbeispiel kann der Prozessor derart konfiguriert sein, daß er einen gewissen Grad an Redundanz innerhalb der einzelnen Ausführungskerne im geteilten Modus behält. Beispielsweise können die Ausführungsressourcen jedes Ausführungskerns für Operationen im geteilten Modus logisch organisiert sein als redundante Ausführungscluster. Die Ausführungscluster bearbeiten den gleichen Befehl im Lock-Step bzw. Verriegelungsschritt, wenn sich der Prozessor im geteilten Modus befindet

und Ergebnisse von den redundanten Clustern können verglichen werden, um weiche Fehler innerhalb eines einzelnen Ausführungskerns zu erfassen.

Figur 1 zeigt ein Blockschaltbild eines Ausführungsbeispiels eines erfindungsgemäßen Computersystems 100. Das Computersystem 100 enthält einen Prozessor 102, einen nicht flüchtigen Speicher 104, einen dynamischen Speicher 106 und eine Systemlogik 108. Die Systemlogik 108 leitet Datenübertragungen zwischen dem Prozessor 102, dem nicht flüchtigen Speicher 104 und dem dynamischen Speicher 106 weiter. Eine Fehlerbehebungsroutine 190 ist in dem nicht flüchtigen Speicher 104 gespeichert, obwohl Teile der Routine 190 in den dynamischen Speicher 106 kopiert (schattenverarbeitet - shadowed) werden können.

Das beschriebene Ausführungsbeispiel des Prozessors 102 enthält einen ersten Ausführungskern 110(a), einen zweiten Ausführungskern 110(b), entsprechende Kernstatusregister (core status register - CSRs) 120(a), 120(b) und eine Prüfeinheit 130. Jedes CSR 120(a), 120(b) enthält ein Kernstatusbit (core status bit - CSB) 124(a) bzw. 124(b). Eine Busschnittstelle 140 zur Daten/Befehlsübertragung zu und von dem Prozessor 102 ist ebenfalls dargestellt. Jeder Ausführungskern 110(a), 110(b) enthält Ressourcen, um Befehle abzurufen (fetch), zu decodieren, auszuführen und zu verabschieden bzw. abzulegen (retire). In der folgenden Beschreibung sind die Bezugnahmen auf die Ausführungskerne 110(a), 110(b) nicht mit einem Index versehen, sofern die Kommentare nicht für einen speziellen Ausführungskern 110 gelten. Bezugnahmen auf die CSRs 120, die CSBs 124 und alle weiteren Ressourcen, die in den Ausführungskernen 110 doppelt vorliegen, werden in ähnlicher Weise behandelt.

Im redundanten Modus führen die Ausführungskerne 110 die gleichen Befehle von einem Codesegment im Lock-Step bzw. Verriegelungsschritt aus, und die Ergebnisse werden von der Prüfeinheit 130 verglichen, um Fehler in einem Ausführungs-

kern 110 zu erfassen. Im geteilten Modus arbeiten die Ausführungskerne 110 "unabhängig". Das heißt, jeder kann verschiedene Befehle von einem oder von mehreren Codesegmenten ausführen. Wie zuvor erwähnt, kann der geteilte Modus einen programmierbaren Hochleistungsmodus für ausgewählte Programme zur Verfügung stellen, da die in dem Prozessor verfügbaren Ausführungsressourcen im geteilten Modus effektiv verdoppelt sind.

Der Unabhängigkeitsgrad zwischen den Ausführungskernen 110 kann im geteilten Modus bei verschiedenen Ausführungsformen des Prozessors 102 variieren. Bei einem Ausführungsbeispiel kann der Prozessor 102 als ein auf einem einzigen Prozessorchip implementiertes SMP-System arbeiten, wenn er sich im geteilten Modus befindet. In diesem Fall arbeitet jeder Ausführungskern 110 als separater Prozessor, der nur den Prozessorchip und bestimmte Komponenten eines Speichersystems mit anderen gemeinsam nutzt. Diese Ausführungsform des Prozessors 102 ist besonders vorteilhaft für einen Code, welcher einen Hochleistungsprozessor erfordert. Eine andere Ausführungsform des Prozessors 102 kann einen gewissen Kopplungsgrad zwischen den Ausführungskernen 110 im geteilten Modus unterstützen, indem geeignete Kanäle zur gemeinsamen Nutzung von Prozessorzustandsinformationen oder von Ausführungskernressourcen vorgesehen werden.

Das Umschalten zwischen dem redundanten und dem geteilten Modus kann unter der Steuerung von Hardware, Software oder Firmware erfolgen. Bei einer Software - und Firmware gesteuerten Umschaltung können Modusumschaltbefehle von den verschiedenen ausgeführten Programmteilprozessen, einem Betriebssystem(operating system - OS)-Scheduler bzw. -Steuerprogramm, Unterbrechungsbehandlern, einer Firmwareroutine oder von ähnlichen Quellen geliefert werden. Bei Hardware gesteuerten Übergängen kann ein Umschalten in Abhängigkeit von erfaßten Bedingungen, zum Beispiel der Ausführung eines bestimmten Befehlstyps, der Erfassung einer Diskrepanz zwi-

schen Ergebnisses in verschiedenen Ausführungskernen oder der Auflösung der Diskrepanz, ausgelöst werden.

Bei einem erfindungsgemäßen Ausführungsbeispiel geben die CSBs 124(a) und 124(b) die Zustände der Ausführungskerne 110(a) bzw. 110(b) an und die CSBs 124(a) und 124(b) geben zusammen den Modus an, in welchem der Prozessor 102 arbeitet. Der Ausführungsmodus des gesamten Prozessors kann mit Hilfe eines einzigen Prozessorstatusbits (processor status bit - PSB) 128 verfolgt werden, auf das von dem mit gestrichelten Linien dargestellten Kästchen in Figur 1 hingewiesen wird. Bei einem Ausführungsbeispiel wird das CSB 124 auf einen ersten Wert, z. B. Eins, gesetzt, wenn der Prozessor 102 im redundanten Modus arbeiten soll, und wird auf einen zweiten Wert, z. B. Null, gesetzt, wenn der Prozessor 102 im geteilten Modus arbeiten soll. Das CSB 124 kann angepaßt werden, wenn ein Modusumschaltbefehl das Umschalten zwischen dem redundanten und dem geteilten Modus auslöst. Verschiedene Ressourcen im Prozessor 102 ändern ihre Operationsweise entsprechend ihrem Ausführungsmodus. Bei der beschriebenen Ausführungsform des Prozessors 102 wird das Abrufen von Befehlen, das Verabschieden von Befehlen und das Prüfen von Ergebnissen im redundanten Modus und im geteilten Modus unterschiedlich gehandhabt, und die entsprechenden Ressourcen passen ihre Operationsweise an die Zustände der CSBs 124 (oder des PSB 128) an.

Eine Ausführungsform des Prozessors 102 mit zwei Ausführungsmoden ist in der US-Patentanmeldung Nr. _____, mit dem Titel "Mikroprocessor With High Reliability Operating Mode" beschrieben, die am gleichen Tage wie die vorliegende Anmeldung eingereicht wurde.

Figur 2A zeigt eine Ausführungsform des Prozessors 102 detaillierter. Bei der dargestellten Ausführungsform ist jeder Ausführungskern 110 als Serie von Stufen in einer Befehlsausführungspipeline dargestellt. Jede Stufe entspricht einer oder mehreren Operationen, die von den Ausführungsker-

nen 110 zum Ausführen ihrer Befehle implementiert sind. Alternativ sind die Pipelinestufen so zu verstehen, daß sie die Logik repräsentieren, welche die angegebene Operation ausführt. Befehle und Daten werden den Ausführungskernen 110 von einem Speichersystem 270 geliefert. Das Speichersystem 270 stellt den dynamischen Speicher 106 und beliebige dazwischenliegende Cache-Speicher dar. Beispielsweise stellt der Cache-Speicher 280 einen Teil des Speichersystems 270 dar, in den Ergebnisse von ausgeführten Befehlen geschrieben werden. Der Cache-Speicher 280 kann auf dem gleichen Chip wie der Prozessor 100 angeordnet sein oder er kann auf einem anderen Chip angeordnet sein.

Bei der beschriebenen Ausführungsform des Prozessors 102 ist jeder Ausführungskern 110 unterteilt in eine Abruf(fetch-FET)-Stufe 210, eine Decodier(DEC)-Stufe 220, eine Register(REG)-Stufe 230, eine Ausführ(execute-EXE)-Stufe, eine Erfasse(detect-DET)-Stufe 250 und eine Verabschiedungs(retirement-RET)-Stufe 260. In der FET-Stufe 210 wird ein Befehl oder werden mehrere Befehle von dem Speichersystem 270 abgerufen. Die abgerufenen Befehle werden in der DEC-Stufe 220 in Mikrooperationen (µops) decodiert und von der bzw. von den Mikrooperationen spezifizierte Quelloperanden werden in der REG-Stufe 230 abgerufen. Die Mikrooperation(en) wird (werden) an den abgerufenen Operanden in der EXE-Stufe 240 ausgeführt, und von der (den) Mikrooperation(en) verursachte Ausnahmen werden in der DET-Stufe 250 gemeldet. Die Mikrooperation(en) wird (werden) in der RET-Stufe 260 verabschiedet, wenn keine Ausnahmen erfaßt werden. Bei der beschriebenen Ausführungsform werden die Ergebnisse von (einer) verabschiedeten Mikrooperation(en) über den Verabschiedungskanal (retirement channel) 264 in den Cache-Speicher 280 geschrieben.

In dieser Beschreibung werden die Begriffe Befehl, Befehlsbündel und Makrobefehl untereinander austauschbar verwendet, genauso wie Mikrooperation und Befehlssilbe. Letz-

tere bezeichnen von den Ausführungseinheiten des Prozessors erkannte Befehle. Erstere bezeichnen Befehle in derjenigen Form, in welcher sie dem Prozessor geliefert werden. Bei einigen Ausführungsbeispielen kann es einen geringen oder keinen Unterschied zwischen diesen Einheiten geben.

Ausführungsformen des Prozessors 102 können einen Puffer zur Entkoppelung der eingangsseitigen bzw. Front-End-Stufe(n) (FET oder FET und DEC) von den ausgangsseitigen bzw. Back-End-Stufen (DEC, REG, EXE, DET und RET oder REG, EXE, DET und RET) aufweisen. Der Puffer speichert abgerufene (oder abgerufene und decodierte) Befehle vorübergehend. Dies ermöglicht es, daß eingangsseitige bzw. Front-End-Operationen weiterablaufen können, selbst wenn ausgangsseitige bzw. Back-End-Operationen angehalten (stalled) oder anderweitig verzögert sind. Es ermöglicht auch das Fortschreiten von Back-End-Operationen, wenn Front-End-Operationen verzögert sind. Eine Ausführungsform des Prozessors 102 verwendet einen Entkopplungspuffer zum Korrigieren von im redundanten Modus erfaßten Fehlern.

Die vorliegende Erfindung erfordert nicht, daß der Prozessor 102 in eine bestimmte Menge von Pipelinestufen unterteilt wird. Beispielsweise kann eine beschriebene Stufe in zwei oder mehr Stufen unterteilt werden, um Zeitvorgaben zu berücksichtigen oder höhere Prozessortaktgeschwindigkeiten zu ermöglichen. Alternativ können zwei oder mehr Stufen zu einer einzigen Stufe zusammengefaßt werden. Andere Ausführungsformen können einen Entkoppelungspuffer enthalten oder nicht. Noch andere Ausführungsformen können Hardware zum Verarbeiten von Befehlen außerhalb der Reihenfolge (out-off-order) enthalten. Die beschriebene Pipeline liefert nur ein Beispiel dafür, wie Operationen in einem Prozessor zur Anwendung der vorliegenden Erfindung aufgeteilt werden können.

Für jeden Ausführungskern 110 sind ferner Status/Steuer(status/control - S/C)-Register 234, Datenregister 238 und ein Daten-Cache-Speicher 244 dargestellt. Die S/C-

Register 234 speichern Informationen, die die Operationsweise des Ausführungskerns 110 regeln. Beispielsweise enthalten die S/C-Register 234 üblicherweise das CSR 120 (und das CSB 124). Die Datenregister 238 speichern zur Verwendung von verschiedenen Ressourcen in dem Ausführungskern 110 vorgesehene Operanden, und der Daten-Cache-Speicher 244 führt eine Zwischenspeicherung von Operanden zwischen dem Speichersystem 270 und anderen Ressourcen in dem Ausführungskern 110 aus. Je nach den Zeitvorgaben kann der Daten-Cache-Speicher 244 Operanden an die Datenregister 238, an Ausführungsressourcen in der EXE-Stufe 240 oder an beide liefern. Bei einer erfindungsgemäßen Ausführungsform stellt jeder Ausführungskern 110 eine Art von Paritätsschutz für die S/C-Register 234, die Datenregister 238 und den Cache-Speicher 244 zur Verfügung.

Die Ausführungskerne 110(a) und 110(b) sind synchronisiert und bearbeiten identische Befehlen im Lock-Step bzw. Verriegelungsschritt, wenn sich der Prozessor 102 im redundanten Modus befindet. Im geteilten Modus können die Ausführungskerne 110(a) und 110(b) verschiedene Befehle unabhängig bearbeiten. Wie zuvor erwähnt, können verschiedene Ausführungsformen des Prozessors 102 verschiedene Grade der Koordinierung zwischen den Ausführungskernen 110(a) und 110(b) im geteilten Modus unterstützen, wie von dem gestrichelt dargestellten Pfeil in Figur 2A angezeigt wird. Wenn der Prozessor 102 beispielsweise im geteilten Modus als Ein-Chip-SMP-System arbeitet, wird eine Koordinierung zwischen den Ausführungskernen 110(a) und 110(b) hauptsächlich während der Modenumschaltvorgänge benötigt. Bei anderen Ausführungsformen des Prozessors 102 können die Ausführungskerne 110(a) und 110(b) Prozesse behandeln, welche eng gekoppelt sind. Diese Ausführungsformen unterstützen eine gemeinsame Benutzung von Daten durch die S/C-Register 234(a) und 234(b), die Datenregister 238(a) und 238(b) und die Daten-Cache Speicher 244(a) und 244(b), sowie eine gewisse Koordi-

nierung der Operationen zwischen den verschiedenen Pipelinestufen.

Figur 2B zeigt eine Ausführungsform der FET-Stufen 210(a), 210(b), die geeignet ist, um den Ausführungskernen 110(a) bzw. 110(b) Befehle im redundanten Modus und im geteilten Modus zu liefern. Jede FET-Stufe 210 enthält einen Befehlszeiger(instruction pointer - IP)-Auswahl-MUX 212 und einen Befehls-Cache-Speicher 214, der mit der DEC-Stufe 220 gekoppelt ist. Die S/C-Register 234 enthalten ein IP-Register 236, das mit Hilfe von Software derart initialisiert werden kann, daß es den nächsten auszuführenden Befehl anzeigt. Außerdem enthält die FET-Stufe 210(b) einen MUX 216 am Ausgang des Cache-Speichers 214(b). Der MUX 216 wird von den CSBs 124 über das UND-Gatter 218 gesteuert. Bei einem erfindungsgemäßen Ausführungsbeispiel kann der Befehls-Cache-Speicher 214 mit Hilfe eines paritätsabhängigen Schemas wie ECC geschützt werden.

Der MUX 212 empfängt IPs an seinen Dateneingängen von verschiedenen Quellen, einschließlich des IP-Registers 236. In Abhängigkeit von einem Signal an seinem Steuereingang wählt der MUX 212 einen IP, der den nächsten aus dem Cache-Speicher 214 abzurufenden Befehl anzeigt. Im geteilten Modus sind die CSBs 124 auf Null gesetzt und der MUX 216 überträgt den von dem Befehls-Cache-Speicher 214(b) gelieferten Befehl. In diesem Modus werden die IP-Register 236(a) und 236(b) unabhängig voneinander initialisiert und aktualisiert, und die Cache-Speicher 214(a) und 214(b) führen entsprechende Befehle den DEC-Stufen 220(a) bzw. 220(b) zu. Im redundanten Modus sind die CSBs 124 auf Eins gesetzt und der MUX 216 liefert den Befehl vom Cache 214(a) an die DEC-Stufe 220(b).

Eine alternative Ausführungsform der FET-Stufen 210 verwendet keinen MUX 216. Statt dessen werden die IP-Register 236(a), 236(b) und die Cache-Speicher 214(a), 214(b) für den redundanten Modus auf den gleichen Zustand initialisiert,

und die FET-Stufen 210, einschließlich der Cache-Speicher 214, arbeiten im Lock-Step bzw. Verriegelungsschritt. Der Fachmann auf dem Gebiet des Prozessordesigns, dem diese Beschreibung zugute kommt, wird weitere Variationen der FET-Stufen 210 erkennen, die verwendet werden können, um für die Ausführungskerne 110 ein unabhängiges und Lock-Step bzw. Verriegelungsschritt-Befehlsabrufen zu implementieren, entsprechend dem Ausführungsmodus, in dem der Prozessor 102 arbeitet.

Figur 2C zeigt ein Blockschaltbild, das eine erfindungsgemäße Ausführungsform der Prüfeinheit 130 darstellt. Die dargestellte Ausführungsform der Prüfeinheit 130 enthält "n" Komparatoren 290(1) bis 290(n), ein ODER-Gatter 294 und ein UND-Gatter 298. Für jede Ausführungseinheit in einem Ausführungskern 110 ist ein Komparator 290 vorgesehen. Beispielsweise kann eine Ausführungsform des Prozessors 102 eine Ganzzahlausführungseinheit (integer execution unit-EU), eine Gleitkommaausführungseinheit (floating point execution unit-FPU), eine Speicherausführungseinheit (memory execution unit-MEU) und eine Verzweigungsausführungseinheit (branch execution unit-BRU) in der EXE-Stufe jedes Ausführungskerns 110 enthalten. Bei dieser Ausführungsform enthält die Prüfeinheit 130 vier Komparatoren 290. Die Komparatoren 290(1), 290(2), 290(3) und 290(4) überwachen die Ausgänge der IEUs, FPUs, MEUs bzw. BRUs von den Ausführungskernen 110(a) und 110(b).

Bei der beschriebenen Ausführungsform der Prüfeinheit 130 erzeugt jeder Komparator 290 einen logischen Wert von Null, wenn die an seine Eingänge angelegten Ergebnisse übereinstimmen, und einen logischen Wert von Eins, wenn die Ausführungsergebnisse nicht übereinstimmen. Bei einer Ausführungsform der Prüfeinheit 130 sind die Komparatoren 290 Selbstprüfkomparatoren (self-check comparators). Das ODER-Gatter 294 erzeugt einen logischen Wert von Eins, wenn irgendeiner der Komparatoren 290 anzeigt, daß seine entspre-

chenden Ausführungsergebnisse nicht übereinstimmen. Das Ausgangssignal des ODER-Gatters 294 dient als FEHLER-Signal, wenn das UND-Gatter 298 aktiviert ist. Bei der beschriebenen Ausführungsform ist dies der Fall, wenn beide CSBs 124 auf Eins gesetzt sind, d. h., wenn sich der Prozessor 102 im redundanten Modus befindet.

Der Fachmann auf dem Gebiet des Prozessordesigns, dem die vorliegende Beschreibung zugute kommt, erkennt weitere Abwandlungen der Prüfeinheit 130, die aktiviert werden kann, um Ergebnisse in den Ausführungskernen 110 zu überwachen, wenn sich der Prozessor 102 im redundanten Modus befindet.

Die vorliegende Erfindung stellt einen Firmwarebasierten Wiederherstellungsmechanismus zur Verfügung, der alleine oder in Kombination mit Hardware- und Software basierten Wiederherstellungsmechanismen realisiert werden kann, um den Prozessor wieder in einen fehlerfreien Zustand zu versetzen, wenn eine Diskrepanz im redundanten Modus erfaßt wird. Es ist unwahrscheinlich, daß die weichen Fehler, denen die vorliegende Erfindung gilt, in beiden Ausführungskernen 110 gleichzeitig auftreten. Ein von der Prüfeinheit 130 erfaßter Unterschied zwischen Ausführungsergebnissen ist wahrscheinlich auf einen weichen Fehler in der Schaltung einer der beiden Ausführungskerne 110 zurückzuführen. Speicherstrukturen wie Registerdateien, Cache-Speicher, Latch-Schaltungen und dergleichen sind besonders anfällig für diese Fehler. Diese Strukturen speichern Spannungspegel, welche Operandenwerte, Befehle oder Mikrooperationen an verschiedenen Punkten der Befehlspipeline repräsentieren. Die vorliegende Erfindung stellt einen Mechanismus zur Verfügung, um die Integrität von Ausführungskernen eines Prozessors wiederherzustellen, welcher im redundanten Modus arbeitet, wenn eine Diskrepanz zwischen von den Ausführungskernen erzeugten Ergebnissen erfaßt wird.

Gemäß einem Ausführungsbeispiel der vorliegenden Erfindung wird auf eine Firmware basierte Wiederherstellungsrou-

tine zugegriffen, wenn eine Diskrepanz zwischen den von den beiden Ausführungskernen des Prozessors erzeugten Ergebnissen im redundanten Modus erfaßt wird. Die Fehlerbehebungsroutine bzw. Wiederherstellungsroutine schaltet den Prozessor auf den geteilten Modus um, in welchem jeder Ausführungskern Daten von seinen Ausführungsressourcen auf Fehler untersucht und die unverfälschten Daten an einem bestimmten Speicherplatz speichert. Die Fehlerbehebungsroutine initialisiert die Ausführungskerne unter Verwendung der unverfälschten Daten und gibt die Steuerung an den unterbrochenen Programmteilprozeß zurück. Bei einer erfindungsgemäßen Ausführungsform sind die untersuchten Ausführungsressourcen paritätsgeschützte Speicherstrukturen. Es werden genügend Prozessorzustandsdaten aus den unverfälschten Daten wiederhergestellt, um beide Ausführungskerne für die weitere Verarbeitung des unterbrochenen Programmteilprozesses zu initialisieren.

Die Verwendung einer Firmware-basierten Wiederherstellungsroutine ermöglicht es dem Prozessor, höher entwickeltere Wiederherstellungsmechanismen zu implementieren. Diese Mechanismen können sich einem breiteren Spektrum von Fehlern widmen, als es Mikrocode- oder Hardware basierte Wiederherstellungsmechanismen können, die beide durch ihre Auswirkung auf die Chipfläche beschränkt sind. Firmware basierte Wiederherstellungsmechanismen können einfacher aktualisiert werden, um andere Fehler handhaben zu können oder um Verbesserungen in Fehlerbehebungsalgorithmen zu implementieren. Darüber hinaus können Firmware basierte Wiederherstellungsmechanismen Systeme einer höheren Ebene, beispielsweise Firmware der Systemebene oder das Betriebssystem benachrichtigen, um sofern erforderlich, eine geeignete Aktion auszuführen.

Bei einer erfindungsgemäßen Ausführungsform kann auf die Fehlerbehebungsroutine mit Hilfe einer Maschinenprüf(maschine check - MC)-Operation zugegriffen werden. Die

MC-Operation kann von der Prüfeinheit 130 ausgelöst werden, wenn einen Diskrepanz erfaßt wird. Alternativ kann der Prozessor zunächst versuchen, sich der Diskrepanz mit Hilfe eines Hardwarewiederherstellungsmechanismussees zu widmen und die Fehlerbehebungsroutine nur dann auslösen, wenn der Hardwaremechanismus fehlschlägt. Zu den Hardwarewiederherstellungsmechanismen gehören beispielsweise das erneute Steuern des Ausführungskerns (der Ausführungskerne) oder von Teilen des Ausführungskerns (der Ausführungskerne), um den Befehlsstrom beginnend mit demjenigen Befehl, für welchen die Diskrepanz erfaßt wurde, erneut auszuführen.

Figur 3 zeigt ein Ablaufdiagramm, das ein erfindungsgemäßes Verfahren 300 zum Korrigieren von weichen Fehlern darstellt. Das Verfahren 300 wird begonnen, wenn ein Fehler erfaßt wird 310, während der Prozessor im redundanten Modus arbeitet. Der Fehler kann durch eine Diskrepanz zwischen den von dem ersten und dem zweiten Ausführungskern erzeugten Ergebnissen angezeigt werden. Wenn der Fehler erfaßt wird 310, springt 320 der Prozessor in eine Fehlerbehebungsroutine (error recovery routine - ERR). Dies kann beispielsweise dadurch geschehen, daß eine MC-Operation ausgelöst wird und der Prozessor mit Hilfe einer zugehörigen Vektortabelle zu der ERR geleitet wird.

Die ERR schaltet 330 den Prozessor von dem redundanten Modus (RM) auf den geteilten Modus (split mode - SM) um. Bei einer Ausführungsform des Verfahrens 300 arbeitet jeder Ausführungskern im geteilten Modus, um unverfälschte Daten aus seinen paritätsgeschützten Speicherstrukturen an einem angegebenen Speicherplatz zu sichern 340. Unverfälschte Daten können dadurch identifiziert werden, daß die Daten in paritätsgeschützten Ressourcen nach Paritätsfehlern abgefragt bzw. auf Paritätsfehler abgetastet (scanning) werden oder daß die Daten von diesen Ressourcen an einen Speicherplatz kopiert werden und die kopierten Daten auf Paritätsfehler abgetastet werden. Wenn beide Ausführungskerne ihre Daten-

sicheroperationen beenden 350, werden die gesicherten Daten abgestimmt 360 und liefern einen unverfälschten Satz von Prozessorzustandsdaten. Bei einer anderen Ausführungsform des Verfahrens 300 sichert 340 nur der Ausführungskern, der keine verfälschten Daten in seinen Speicherstrukturen identifiziert, seine Daten an dem angegebenen Speicherplatz. Der Prozessor kehrt in den redundanten Modus zurück 370, die Ausführungskerne werden mit Hilfe der wiederhergestellten Prozessorzustandsdaten initialisiert 380 und die Ausführung des unterbrochenen Teilprozesses wird wieder aufgenommen 380.

Bei einer erfindungsgemäßen Ausführungsform wird der Prozessor 102 zwischen dem redundanten Modus und dem geteilten Modus unter Softwaresteuerung, z. B. mit Hilfe eines Modusumschaltbefehls, umgeschaltet. Bei einer anderen erfindungsgemäßen Ausführungsform kann der Prozessor 102 mit Hilfe eines Hardwaremechanismussees umgeschaltet werden.

Bevor eine Diskrepanz erfaßt wird, befindet sich der Prozessor im redundanten Modus und der Ausführungskern arbeitet im Lock-Step bzw. Verriegelungsschritt. Wenn eine Diskrepanz erfaßt wird, kann die Fehlerbehebungsroutine die Modusumschaltung dadurch realisieren, daß ein Schalte-Auf-Geteilten-Modus-Um-Befehl (switch-to-redundant-mode - SW_RM) an jeden Ausführungskern geliefert wird. Im redundanten Modus werden diese Befehle in den Pipelinestufen der beiden Ausführungskerne im Lock-Step bzw. Verriegelungsschritt abgearbeitet (staged down), wobei der Prozessor auf den geteilten Modus umgeschaltet wird, wenn sie verabschiedet werden. Folglich ist es nicht erforderlich, die Ausführungskerne beim Umschalten auf den geteilten Modus zu synchronisieren. Wenn sich der Prozessor im geteilten Modus befindet, kann jeder Ausführungskern die Datenwiederherstellungsoperationen der ERR unabhängig ausführen.

Wenn der Ausführungskern (die Ausführungskerne) ihre Datenwiederherstellungsoperationen beenden, kehrt der Prozes-

sor von dem geteilten Modus wieder in den redundanten Modus zurück, um den unterbrochenen Programmteilprozeß fortzusetzen. Bei einer erfindungsgemäßen Ausführungsform kann dieser Übergang in Stufen dadurch realisiert werden, daß ein Schalte-Auf-Redundanten-Modus-Um-Befehl (switch-to-redundant-mode - SW_RM) in jedem Ausführungskern ausgeführt wird. Die mehrstufige Realisierung des Übergangs ermöglicht eine Anpassung bei Unterschieden in den Zeitpunkten, zu denen die Ausführungskerne ihre Datensicheroperationen beenden.

Bei der beschriebenen Ausführungsform schaltet der SW_RM-Befehl in einem ersten Schritt einen Ausführungskern auf einen "Bereitzustand (ready state)" um, z. B. CSB(b) 124 = 1. In dem zweiten Schritt schaltet der SW_RM-Befehl den anderen Ausführungskern auf einen "Bereitzustand" um, z. B. CSB (a) = 1. Der Prozessor kehrt in den redundanten Modus zurück, wenn beide Ausführungskerne den Bereitzustand erreichen, z. B. PSB = CSB(a) UND CSB(b). In Abhängigkeit von dem zur Ablaufsteuerung der Modusumschaltbefehle verwendeten Algorithmen können sich die erste und zweite Stufe vollständig überlappen (gleichzeitiges Umschalten), teilweise überlappen oder gar nicht überlappen (sequentielles Umschalten).

Der Bereitzustand ermöglicht es, daß die Ausführungskerne synchronisiert sind, bevor der Prozessor von dem geteilten Modus in den redundanten Modus umschaltet. Bei anderen erfindungsgemäßen Ausführungsformen kann die Synchronisierung mit Hilfe eines Hardwaremechanismussees realisiert werden. Beispielsweise kann der erste Ausführungskern in einen Bereitzustand übergehen, wenn eine Ende-der-Routine-Bedingung erfaßt wird, und der zweite Ausführungskern kann in den Bereitzustand übergehen, wenn der erste Ausführungskern seinen Übergang beendet. Wie bei den Software gesteuerten Modusumschaltungen erlaubt es der Bereitzustand den Ausführungskernen, sich zu treffen, bzw. anzunähern (rendezvous), bevor der Prozessor in den redundanten Modus zurückgeschaltet wird.

Figur 4 zeigt eine Ausführungsform eines von einem Ausführungskern implementierten Verfahrens 400 zum Sichern unverfälschter Daten. Beispielsweise kann das Verfahren 400 einen Teil der ERR bilden, die von einem Ausführungskern nach dem Umschalten in den geteilten Modus implementiert wird. Wie oben erwähnt, kann die ERR mit Hilfe einer Maschinenprüfoperation erreicht werden, welche die Steuerung des Prozessors mit Hilfe einer zugeordneten Vektortabelle auf die ERR überträgt. Die ERR schaltet den Prozessor von dem redundanten Modus auf den geteilten Modus um, wodurch es jedem Ausführungskern ermöglicht wird, auf die in Figur 4 dargestellte Datenprüf- und Wiederherstellungsroutine unabhängig zuzugreifen und diese unabhängig zu implementieren.

Gemäß dem beschriebenen Verfahren werden unkritische Speicherstrukturen geräumt (flushed) 410. Zu den unkritischen Speicherstrukturen gehören beispielsweise Cache-Speicher, Register und Speicherstrukturen, welche keine architektonischen Zustandsdaten speichern. Diese Speicherstrukturen sind üblicherweise nicht paritätsgeschützt. In Abhängigkeit von der Prozessorausführungsform können diese Strukturen L0-Befehls- und Daten-Cache-Speicher, Verzweigungszielpuffer, Vorgerücktes-Laden-Verfolgungstabellen und dergleichen sein.

Der Inhalt der Datenregisterdateien bzw. -registersätze wird auf Paritätsfehler geprüft 420. Wenn kein Paritätsfehler gefunden wird 430, wird der Inhalt der Datenregisterdateien an einen bestimmten Speicherplatz kopiert 440. Wie oben erwähnt, kann dies ein Speicherplatz sein, der mit dem anderen Ausführungskern gemeinsam genutzt wird, oder es kann ein für den speziellen Ausführungskern reservierter Speicherplatz sein. Wenn ein Paritätsfehler gefunden wird 430, werden keine Daten aus den Datenregisterdateien kopiert. Alternativ können lediglich unverfälschte Daten aus der (den) Datenregisterdatei(en) kopiert werden.

Die C/S-Registerdateien werden ferner auf Paritätsfehler geprüft 450. Zu diesen gehören beispielsweise die oben beschriebenen Kernstatusregister, verschiedene Register, welche Informationen über den aktuellen Teilprozeß speichern, und die Übersetzungsregister in assoziativen Übersetzungspufferspeichern (translation look-aside buffers - TLB). Wenn in den C/S-Registerdateien kein Paritätsfehler gefunden wird 460, wird der Inhalt der C/S-Registerdateien an einen bestimmten Speicherplatz kopiert 570. Wenn ein Paritätsfehler gefunden wird, werden keine Daten aus der (den) C/S-Registerdatei(en) kopiert. Alternativ können lediglich unverfälschte Daten von der (den) C/S-Registerdatei(en) kopiert werden.

Ausführungsformen des Prozessors können Fehlerkorrekturcodes (error correction codes - ECC) für ausgewählte Cache-Speicher firmwaremäßig implementieren. Dadurch wird keine ECC-Korrektur-Hardware im Prozessor mehr benötigt, obwohl die Cache-Speicher die ECC-Bits weiterhin speichern. Bei diesen Ausführungsformen können 1- und 2-Bit-Fehler mit der ERR korrigiert werden 480. Das Kästchen 480 wird umgangen, wenn der ECC hardwaremäßig implementiert ist. Wenn alle erfaßten Fehler korrigiert wurden, springt der Ausführungskern in einen Wartezustand 490.

Wenn jeder Ausführungskern das Verfahren 400 ausgeführt hat, enthält der bestimmte bzw. spezifizierte Speicherplatz nur unverfälschte Daten. Eine Folge der statistischen Natur von weichen Fehlern ist, daß es sehr unwahrscheinlich ist, daß der gleiche Datenblock in beiden Ausführungskernen gleichzeitig verfälscht ist. Infolge dessen liefert der Datensicherungsprozeß genügend unverfälschte Prozessorzustandsdaten, um die Ausführungskerne zu initialisieren und den unterbrochenen Programmteilprozeß wieder aufzunehmen.

Die vorliegende Erfindung erfordert es nicht, daß die Speicherstrukturen in einer bestimmten Reihenfolge geräumt oder geprüft werden. Beispielsweise können Datenregisterda-

teien nach C/S-Registerdateien geprüft werden und unkritische Dateien können zu jeder passenden Zeit in dem Prozeß geräumt werden. Die oben angegebene Reihenfolge dient lediglich der Veranschaulichung.

Zusätzlich zur Sicherung unverfälschter Daten an einem bestimmten Speicherplatz stellt die ERR einen Mechanismus zur Verfügung, um die gesicherten Daten abzustimmen bzw. in Einklang zu bringen (reconcile)und um jedem Ausführungskern genügend unverfälschte Prozessorzustandsdaten zur Wiederaufnahme des unterbrochenen Programmteilprozesses zu liefern.

Figur 5A zeigt ein Blockschaltbild, das einen Mechanismus zum Abstimmen der von jedem Ausführungskern in einem Prozessor mit zwei Ausführungskernen gesicherten Daten darstellt. Der Ausführungskern 110(b) implementiert die ERR, um seine unverfälschten Daten in einen bestimmten Speicherplatz 510(b) im Speichersystem 500 zu kopieren (1). Der Ausführungskern 110(a) implementiert die ERR, um seine unverfälschten Daten an einen bestimmten Speicherplatz 510(a) zu kopieren (2). Bei dem beispielhaften Mechanismus läßt ein weicher Fehler in den Speicherstrukturen des Ausführungskerns 110(b) einen unvollständigen Satz von Prozessorzustandsdaten zurück. Der Ausführungskern 110(a) welcher keine Paritätsfehler erfahren hat, kopiert (2) einen vollständigen Satz von Prozessorzustandsdaten in den Speicherplatz 510(a). Hier bezeichnet "vollständig" eine Teilmenge der Prozessorzustandsdaten, welche ausreicht, um dem Prozessor 102 die Wiederaufnahme des unterbrochenen Programmteilprozesses zu erlauben. Die verfälschten Daten in den Speicherstrukturen des Ausführungskerns 110(b) können verfolgt werden, wenn der Ausführungskern 110(b) seine unverfälschten Daten in den Speicherplatz 510(b) kopiert (1), um einen Datensatz der Daten aufrecht zu erhalten bzw. zu führen, die aktualisiert werden müssen.

Nach den Datensicheroperationen (1), (2), führt der Ausführungskern 110(b) die ERR aus, um unverfälschte Daten von

dem Speicherplatz 510(a) in den Speicherplatz 510(b) zu kopieren. Dadurch wird ein zweiter vollständiger Satz von Prozessorzustandsdaten zur Neuinitialisierung der Ausführungskerne des Prozessors 102 zur Verfügung gestellt. Wenn der Ausführungskern 110(b) beispielsweise bestimmt, daß der Ausführungskern 110(a) seine Kopieroperationen beendet hat, kann er Daten vom Speicherplatz 510(a) in den Speicherplatz 510(b) nur für diejenigen Datenblöcke kopieren, die verfälschte Daten in seinen zugeordneten Speicherstrukturen entsprechen. Dieser Datenabstimmungsmechanismus (data reconciliation mechanism) liefert zwei Sätze von Prozessorzustandsdaten im Speicher 500, welche dann zum Initialisieren (4) und (5) der Ausführungskerne 110(a) bzw. 110(b) verwendet werden können. Die Initialisierungen (4) und (5) können von jedem Ausführungskern 110 im geteilten Modus getrennt erledigt werden oder sie können von den Ausführungskernen im redundanten Modus gleichzeitig erledigt werden.

Bis auf den Teil der ERR, welcher unverfälschte Daten aus dem Speicherplatz 510(a) in den Speicherplatz 510(b) kopiert, erfolgt der beschriebene Datenabstimmungsmechanismus mit Hilfe von redundanten Operationen. Daten werden von jedem Ausführungskern geprüft, im Speicher gesichert und in Speicherstrukturen zurückgeschrieben. Dies verringert die Wahrscheinlichkeit, daß ein in einem Satz der Prozessorzustandsdaten während der Wiederherstellung erzeugter weicher Fehler beide Ausführungskerne verfälscht.

Figur 6 zeigt ein Ablaufdiagramm, das den Datenabstimmungsmechanismus gemäß Figur 5A darstellt. Dieser entspricht beispielsweise dem Kästchen 490 gemäß Figur 4. Ein den Datenabstimmungsteil der ERR implementierender Prozessor bestimmt 610, ob der andere Ausführungskern seine Datensicheroperationen beendet hat. Wenn diese Operationen beendet wurden, bestimmt 620 der Ausführungskern, ob er seine gesicherten Daten aktualisieren muß.

Bei einer Ausführungsform des Verfahrens 600 aktualisiert 630 der Ausführungskern, der verfälschte Daten in seinen Speicherstrukturen erfaßt hat, seine gesicherten Daten mit Hilfe entsprechender Daten von dem Speicherplatz des anderen Ausführungskerns. Beispielsweise kann der Ausführungskern einen Hinweis auf die Datenblöcke speichern, die während seiner Sicheroperationen verfälscht erscheinen. Dieser Ausführungskern kann diese Hinweise dann verwenden, um unverfälschte Versionen der Daten von der entsprechenden Speicheradresse des bestimmten Speicherplatzes des anderen Ausführungskerns abzurufen. Der Ausführungskern, der keine verfälschten Daten erfaßt hat wartet, bis der andere Ausführungskern seine Aktualisierung 630 beendet 640. Wenn beide Ausführungskerne den Wartezustand 640 erreichen, kehrt der Prozessor 650 in den redundanten Modus zurück und initialisiert 660 die Ausführungskerne mit den abgestimmten Daten. Bei einer alternativen Ausführungsform des Verfahrens 600 können die Ausführungskerne im geteilten Modus initialisiert werden und dann wird in den redundanten Modus zurückgekehrt.

Figur 5B zeigt einen alternativen Mechanismus zur Datenabstimmung. Beide Ausführungskerne 110(a) und 110(b) schreiben (1) bzw. (2) ihre unverfälschten Daten bei diesem Mechanismus an entsprechende Speicheradressen in dem gleichen Speicherplatz 520. Bei einer Ausführungsform des Mechanismus schreibt nur der Ausführungskern seine gesamten Prozessorzustandsdaten in den Speicherplatz 520, der keine verfälschten Daten hat. Der Ausführungskern, der verfälschte Daten in seinen Datenstrukturen erfaßt, kopiert die verfälschten Daten nicht in den Speicherplatz 520.

Eine andere Ausführungsform des Mechanismus gemäß Figur 5B kann Daten von dem Ausführungskern, der in seinen zugeordneten Speicherstrukturen keine verfälschten Daten erfaßt, schreiben und den Sicherungsschritt für den anderen Ausführungskern, d. h., denjenigen Kern, der verfälschte Daten enthält, überspringen. In jedem Fall sind die in den Spei-

cherplatz 520 kopierten Prozessorzustandsdaten vollständig und können an die Ausführungskerne 110(a) bzw. 110(b) zurückgeschrieben werden (3) und (4), um diese für die Wiederaufnahme des unterbrochenen Programmteilprozesses neu zu initialisieren. Da nur ein Satz von Prozessorzustandsdaten bei dem Datenabstimmungsmechanismus gemäß Figur 5B aufrecht gehalten wird, ist ein weicher Fehler in diesem Datensatz während der Wiederherstellung durch einen einfachen Ergebnisvergleich nicht erfaßbar.

Bei den oben beschriebenen erfindungsgemäßen Ausführungsformen wird die Fehlerbehebung implementiert, während der Prozessor im geteilten Modus ist, um es jedem Ausführungskern zu ermöglichen, Fehler in seinen zugeordneten Speicherstrukturen unabhängig zu identifizieren. Weiche Fehler können auch während dieser Fehlerbehebungsoperationen auftreten, wenn eine redundante Ausführung zur Erfassung von weichen Fehlern nicht verfügbar ist. Für hochgradig zuverlässige Systeme kann selbst diese begrenzte Anfälligkeit für weiche Fehler zu groß sein.

Bei einer erfindungsgemäßen Ausführungsform können die Ausführungsressourcen jedes Ausführungskerns derart konfiguriert sein, daß sie beim Betrieb im geteilten Modus als zwei oder mehr logische Cluster parallel operieren. Während jeder Ausführungskern die ERR unabhängig implementiert, implementieren logisch definierte Ausführungscluster innerhalb jedes Ausführungskerns Befehle von der ERR redundant. Bei dieser Ausführungsform werden Befehle von der ERR innerhalb jedes Ausführungskerns vervielfältigt und an verschiedene Ausführungscluster geleitet. Die von den verschiedenen Ausführungsclustern erzeugten Ergebnisse können verglichen werden, um zu bestimmen, ob ein weicher Fehler bei der Verarbeitung der ERR-Befehle auftrat.

Figur 7 zeigt ein Blockschaltbild einer Ausführungsform eines Ausführungskerns 710, der dieses Merkmal der internen Redundanz enthält. Die beschriebene Ausführungsform des Aus-

führungskerns 710 enthält einen Befehls-Cache-Speicher (instruction cache - I-Cache)714, eine Decodier- oder Verteileinheit (dispersal unit) 720, eine Registerdatei (Registerdateien) 730, ein erstes und zweites Ausführungscluster 740(a) bzw. 740(b), eine Prüfeinheit 760 und eine Verabschiedungseiheit 770. In der folgenden Beschreibung sind Bezugnahmen auf die Ausführungscluster 740(a) und 740(b) nicht mit Index versehen, sofern es nicht zur Vermeidung einer Mehrdeutigkeit erforderlich ist. Bezugnahmen auf andere Ressourcen in dem Ausführungskern 710, welche doppelt vorliegen bzw. dupliziert sein können, wie beispielsweise die die Ausführungscluster 740 bildenden Ausführungseinheiten, werden ähnlich behandelt.

Bei der beschriebenen Ausführungsform enthält jedes Ausführungscluster 740 eine Verzweigungsausführungseinheit (branch execution unit - BRU) 752, eine Ganzzahlausführungseinheit (integer execution unit - IEU) 754, eine Speichermanagement-(Lade/Speichere)-Einheit (memory management unit - MMU) 756 und eine Gleitkommaeinheit (floating point unit - FPU) 758 (allgemein „Ausführungseinheiten" genannt). Verschiedene Ausführungsformen des Ausführungskerns 710 können verschiedene Arten und unterschiedlich viele Ausführungseinheiten enthalten, ohne den Schutzbereich der vorliegenden Erfindung zu verlassen. Beispielsweise verwendet der Merced-Prozessor der Intel® Corporation eine Verzweigungsausführungseinheit, die drei separate Verzweigungsausführungseinheiten enthält. Die Ausführungscluster 740 sind getrennt voneinander dargestellt, um deren logische Organisation im geteilten Modus zu veranschaulichen und nicht um eine tatsächliche Trennung zwischen den verschiedenen Ausführungsressourcen wiederzugeben.

Bei anderen Ausführungsformen des Ausführungskerns 710 können nicht alle Ausführungseinheiten 750 dupliziert sein oder Befehle können an bestimmte Ausführungseinheiten nicht dupliziert werden. Beispielsweise benötigen Verzweigungsaus-

führungseinheiten beträchtliche Prozessorressourcen zur Unterstützung, und beträchtliche Chipflächen können dadurch eingespart werden, daß die BRU 752 nicht dupliziert wird. Dies wird von dem durch gestrichelte Linien dargestellten Kästchen um die BRU 752(b) in Figur 7 herum angezeigt. In diesem Fall können Verzweigungsbefehle nicht redundant ausgeführt werden, oder es können andere Mechanismen zum Prüfen ihrer Ausführung verwendet werden. Beispielsweise können Verzweigungsbefehle dupliziert und seriell ausgeführt werden. In ähnlicher Weise kann das Duplizieren von Lade- und Speicherbefehlen Bandbreite des Speichersystems im HR-Modus belegen. In diesem Fall können mehrere MMUs 756 implementiert werden, um mehrere Lade- und/oder Speicheroperationen in einem Befehlsbündel zu ermöglichen, aber die einzelnen Lade-/Speicheroperationen werden bei diesen Befehlen im HR-Modus nicht dupliziert. Ähnliche Regelungen können für andere Ausführungseinheiten 750 und Befehlstypen verwendet werden, ohne den Erfindungsgedanken zu verlassen.

Der I-Cache-Speicher 714 liefert der Decodiereinheit 720 Befehle, welche diese über eine oder mehrere Registerdateien 730 an geeignete Ausführungseinheiten weiterleitet. Zu den Registerdateien 730 gehören Datenregisterdateien 732, Steuer/Status(controle/status - C/S)-Register 734 und eine Registerumbenennungseinheit 738. Die Datenregisterdateien 732 ermöglichen eine temporäre Speicherung für beispielsweise Ganzzahl- und Gleitkommaoperanden, welche von den Ausführungseinheiten 750 bearbeitet werden. Die Umbenennungseinheit 738 bildet die in Mikrooperationen spezifizierten virtuellen Registerbezeichner bzw. -identifizierer auf die physikalischen Register in der Registerdatei 730 ab.

Die C/S-Register 734 speichern Informationen, welche die Arbeitsweise des Ausführungskerns 710 steuern, und den Status der verschiedenen Ausführungsressourcen. Bei einer Ausführungsform des Ausführungskerns 710 liefert die Ausgabeeinheit 720 identische Befehle an die Ausführungscluster

740(a) und 740(b), wenn sich der Prozessor im geteilten Modus befindet. Beispielsweise liefert die Ausgabeeinheit 720 die Mikrooperationen von einem Befehl (oder die Mikrooperationen von identischen Befehlen) an geeignete Ausführungseinheiten in den Ausführunsclustern 740(a) und 740(b). Die von den Clustern 740(a) und 740(b) erzeugten Ergebnisse werden von der Prüfeinheit 760 verglichen, und es wird ein Fehler angezeigt, wenn sich die Ausführungsergebnisse unterscheiden. Im redundanten Modus können die Ausführungscluster 740(a) und 740(b) unabhängig voneinander gesteuert werden und verschiedene Befehle verarbeiten. Die Redundanz wird dadurch erzeugt, daß auf dem anderen Ausführungskern identische Befehle ausgeführt werden. Der Ausführungskern 710 ermöglicht eine Redundanz auf Ausführungsclusterebene, um einen Prozessor beim Betrieb im geteilten Modus vor weichen Fehlern zu schützen. Eine Ausführungsform eines Prozessors, welcher einen Ausführungskern 710 enthält, wird ausführlicher in der US-Patentanmeldung Nr. _____ mit dem Titel „Mikroprocessor With High Reliability Operating Mode" beschrieben, welche am gleichen Tage wie die vorliegende Anmeldung eingereicht wurde.

Es wurde somit ein Firmware-basierter Mechanismus zum Korrigieren von weichen Fehlern in einem Prozessor mit zwei Kernen beschrieben, welcher zwischen einem redundanten Ausführungsmodus und einem geteilten Ausführungsmodus hin und her geschaltet werden kann. Wenn ein Fehler bei der Ausführung eines Programmteilprozesses im redundanten Modus erfaßt wird, schaltet der Prozessor in den geteilten Modus um. Im geteilten Modus identifiziert ein Ausführungskern unverfälschte Prozessorzustandsdaten in seiner Speicherstruktur und kopierte diese unverfälschten Daten an einen bestimmten Speicherplatz. Die unverfälschten Daten werden dann derart abgestimmt, daß sie ausreichend Prozessorzustandsdaten zur Wiederaufnahme der Ausführung des unterbrochenen Programmteilprozesses liefern. Der Prozessor kehrt in den redundan-

ten Ausführungsmodus zurück, die Ausführungskerne werden mit den Prozessorzustandsdaten initialisiert, und der unterbrochene Programmteilprozeß wird wieder aufgenommen. Das Umschalten zwischen dem redundanten Modus und dem geteilten Modus kann mit Hilfe von Modusumschaltbefehlen realisiert werden.

Die beschriebenen Ausführungsformen wurden zur Veranschaulichung verschiedener Merkmale der vorliegenden Erfindung geliefert. Der Fachmann auf dem Gebiet des Prozessordesigns, dem diese Beschreibung zugute kommt, wird Abwandlungen und Modifikationen der beschriebenen Ausführungsformen erkennen, welche dennoch unter den Erfindungsgedanken und Schutzbereich der beigefügten Ansprüche fallen.

PCT/US00/41079

INTEL CORPORATION
2200 Mission College Boulevad
Santa Clara, Calivornia 95052
U.S.A.

FIRMWAREMECHANISMUS ZUM KORRIGIEREN VON WEICHEN FEHLERN

Zusammenfassung

Ein Computersystem enthält einen Prozessor mit zwei Ausführungskernen und einem nicht flüchtigen Speicher, der eine Fehlerbehebungsroutine speichert. Die Ausführungskerne des Prozessors arbeiten im Lock-Step, wenn sich der Prozessor in einem redundanten Ausführungsmodus befindet und sie arbeiten voneinander unabhängig, wenn sich der Prozessor in einem geteilten Ausführungsmodus befindet. Die Fehlerbehebungsroutine wird aufgerufen, wenn der Prozessor beim Betrieb in dem redundanten Ausführungsmodus einen weichen Fehler erfaßt. Die Fehlerbehebungsroutine schaltet den Prozessor in den geteilten Ausführungsmodus um. Im geteilten Modus sichert jeder Ausführungskern unverfälschte Prozessorzustandsdaten an einem bestimmten Speicherplatz und aktualisiert verfälschte Daten mit entsprechenden Prozessorzustandsdaten von dem anderen Ausführungskern. Die Fehlerbehebungsroutine bringt den Prozessor in den redundanten Modus zurück, initialisiert jeden Ausführungskern mit den wiederhergestellten Prozessorzustandsdaten und gibt die Steuerung des Prozessors an den Programmteilprozessor zurück, der bei der Erfassung des weichen Fehlers ausgeführt wurde.

(Fig. 3)

PATENTANWÄLTE ZENZ, HELBER, HOSBACH & PARTNER · HUYSSENALLEE 58-64 · D-45128 ESSEN

PCT/US00/41079

Patentansprüche

1. Computersystem mit:

einem Prozessor, der einen ersten und einen zweiten Ausführungskern und eine Prüfeinheit aufweist, wobei der erste und der zweite Ausführungskern Befehle unabhängig verarbeiten, wenn sich der Prozessor in einem geteilten Modus befindet, und identische Befehle im Lock-Step verarbeiten, wenn sich der Prozessor in einem redundanten Modus befindet, und wobei die Prüfeinheit Ergebnisse von dem ersten und dem zweiten Ausführungskern vergleicht, wenn sich der Prozessor in dem redundanten Modus befindet; und

mit einem nicht flüchtigen Speicher, in welchem eine Fehlerbehebungsroutine gespeichert ist, welche den Prozessor in den geteilten Modus umschaltet, wenn die Prüfeinheit einen Fehler erfaßt, die unverfälschte Prozessorzustandsdaten aus wenigstens einem der Ausführungskerne identifiziert und den ersten und zweiten Ausführungskern mit den identifizierten Prozessorzustandsdaten initialisiert.

2. Computersystem nach Anspruch 1, wobei die Fehlerbehebungsroutine den Prozessor in den redundanten Modus zurückbringt.

3. Computersystem nach Anspruch 1, wobei die Fehlerbehebungsroutine unverfälschte Prozessorzustandsdaten in jedem Ausführungskern zugeordneten ausgewählten Speicherstrukturen identifiziert, die unverfälschten Prozessorzustandsdaten in einen bestimmten Speicherplatz kopiert und die kopierten Prozessorzustandsdaten in Einklang bringt, wenn der Prozessor in den geteilten Modus eintritt.

4. Computersystem nach Anspruch 3, wobei die Fehlerbehebungsroutine den ersten und zweiten Ausführungskern in einem

Rendezvous-Zustand synchronisiert und den Prozessor in den redundanten Modus zurückbringt.

5. Computersystem nach Anspruch 1, wobei das Computersystem ferner einen Hauptspeicher enthält und die Fehlerbehebungsroutine die Prozessorzustandsdaten in einem nicht Cache-speicherbaren (uncacheable) Teil des Hauptspeichersystems sichert.

6. Verfahren zum Behandeln von weichen Fehlern in einem Prozessor, der einen ersten und einen zweiten Ausführungskern in einem redundanten und einem geteilten Modus arbeiten lassen kann, wobei das Verfahren die Schritte aufweist,

daß ein weicher Fehler erfaßt wird, wenn der Prozessor in dem redundanten Modus arbeitet;

daß eine Fehlerbehebungsroutine auf jedem Ausführungskern ausgeführt wird, um unverfälschte Daten aus den dem ersten und dem zweiten Ausführungskern zugeordneten Speicherstrukturen zu sichern; und

daß Prozessorzustandsdaten aus den gesicherten unverfälschten Daten wiederhergestellt werden.

7. Verfahren nach Anspruch 6, wobei das Ausführen der Fehlerbehebungsroutine die Schritte enthält,

daß der Prozessor von dem redundanten Modus in den geteilten Modus umgeschaltet wird; und

daß unverfälschte Daten von jedem Ausführungskern in einem bestimmten Speicherplatz gesichert werden.

8. Verfahren nach Anspruch 7, ferner mit den Schritten, daß die gesicherten unverfälschten Daten in Einklang gebracht werden, um die Prozessorzustandsdaten wiederherzustellen; und

daß der erste und zweite Ausführungskern unter Verwendung der wiederhergestellten Prozessorzustandsdaten initialisiert werden.

9. Verfahren nach Anspruch 8, wobei das in Einklang bringen der gesicherten unverfälschten Prozessorzustandsdaten die Schritte aufweist,

daß verfälschte Prozessorzustandsdaten in einem der ersten und zweiten Ausführungskerne identifiziert werden; und

daß der dem einen der ersten und zweiten Ausführungskerne zugeordnete bestimmte Speicherplatz unter Verwendung von unverfälschten Prozessorzustandsdaten aus dem bestimmten Speicherplatz des anderen Ausführungskerns der ersten und zweiten Ausführungskerne aktualisiert wird.

10. Maschinenlesbares Medium auf dem Befehle gespeichert sind, die von einem Prozessor mit zwei Ausführungskernen ausgeführt werden können, um ein Verfahren zur Behebung von weichen Fehlern zu implementieren, wobei das Verfahren die Schritte aufweist,

daß der Prozessor beim Erfassen eines weichen Fehlers derart umgeschaltet wird, daß er die beiden Ausführungskerne unabhängig arbeiten läßt;

daß eine Fehlerbehebungsroutine auf jedem Ausführungskern ausgeführt wird, um einen minimalen Satz von Prozessorzustandsdaten aus den unverfälschten Prozessorzustandsdaten wiederherzustellen; und

daß jeder der beiden Ausführungskerne unter Verwendung des minimalen Satzes von Prozessorzustandsdaten initialisiert wird.

11. Maschinenlesbares Medium nach Anspruch 10, wobei das Initialisieren jedes der beiden Ausführungskerne die Schritte aufweist,

daß der Prozessor derart umgeschaltet wird, daß er die beiden Ausführungskerne im Lock-Step arbeiten läßt; und

daß der minimale Satz von Prozessorzustandsdaten in jeden der beiden Ausführungskerne kopiert wird.

12. Maschinenlesbares Medium nach Anspruch 10, wobei das Ausführen einer Fehlerbehebungsroutine die Schritte aufweist,

daß paritätsgeschützte Speicherstrukturen auf jedem Ausführungskern abgetastet werden, um unverfälschte Prozessorzustandsdaten zu identifizieren;

daß die unverfälschten Prozessorzustandsdaten in den jedem Ausführungskern zugeordneten bestimmten Speicherplätzen gesichert werden; und

daß der minimale Satz von Prozessorzustandsdaten aus den gesicherten Prozessorzustandsdaten wiederhergestellt wird.

13. Computersystem, mit:

einem einen ersten und einen zweiten Ausführungskern aufweisenden Prozessor, wobei der erste und zweite Ausführungskern derart ausgebildet sind, daß sie im Lock-Step arbeiten, wenn sich der Prozessor in einem redundanten Modus befindet, und daß sie unabhängig arbeiten, wenn sich der Prozessor in einem geteilten Modus befindet; und

einem nicht flüchtigen Speicher, in dem eine Fehlerbehebungsroutine gespeichert ist, um den Prozessor in den geteilten Modus umzuschalten, wenn ein Fehler im redundanten Modus erfaßt wird, um unverfälschte Prozessorzustandsdaten in jedem Ausführungskern zu identifizieren und den Prozessor unter Verwendung eines Satzes von aus den unverfälschten Prozessorzustandsdaten wiederhergestellten Prozessorzustandsdaten in den redundanten Modus zurückzubringen.

14. Computersystem nach Anspruch 13, wobei die Fehlerbehebungsroutine den Prozessor mit Hilfe eines Modusumschaltbefehls in den geteilten Modus umschaltet.

15. Computersystem nach Anspruch 14, wobei jeder Ausführungskern nach dem Umschalten in den geteilten Modus einen Teil des Fehlerbehebungsprogramms unabhängig ausführt, um seinen Speicherstrukturen zugeordnete unverfälschte Prozessordaten zu identifizieren und an einem bestimmten Speicherplatz zu sichern.

16. Computersystem nach Anspruch 15, wobei einer der Ausführungskerne verfälschte Daten aus seinen Speicherstrukturen unter Verwendung von gesicherten Prozessorzustandsdaten aus dem bestimmten Speicherplatz des anderen Ausführungskerns aktualisiert.

17. Computersystem nach Anspruch 16, wobei die Fehlerbehebungsroutine den Prozessor mit Hilfe eines Modusumschaltbefehls in den redundanten Modus zurückbringt.

18. Computersystem mit:

einem Prozessor, der einen ersten und einen zweiten Ausführungskern aufweist, die im Sperrschritt arbeiten, wenn sich der Prozessor in einem redundanten Ausführungsmodus befindet, und die unabhängig voneinander arbeiten, wenn sich der Prozessor in einem geteilten Ausführungsmodus befindet; und

mit einem nicht flüchtigen Speicher, in dem Befehle gespeichert sind, die von dem Prozessor implementiert werden können, um ein Verfahren zum Beheben von weichen Fehlern auszuführen, wenn sich der Prozessor in dem redundanten Ausführungsmodus befindet, wobei das Verfahren die Schritte aufweist,

daß der Prozessor in einen geteilten Ausführungsmodus umgeschaltet wird;

daß unverfälschte Prozessorzustandsdaten von jedem Ausführungskern an einem bestimmten Speicherplatz gesichert werden;

daß ein minimaler Satz von Prozessorzustandsdaten aus den gesicherten unverfälschten Prozessorzustandsdaten wiederhergestellt wird; und

daß jeder Ausführungskern mit dem minimalen Satz von Prozessorzustandsdaten initialisiert wird.

19. Computersystem nach Anspruch 18, wobei das Wiederherstellen des minimalen Satzes von Prozessorzustandsdaten den Schritt aufweist, daß verfälschte Prozessorzustandsdaten von einem der Ausführungskerne durch unverfälschte Prozessorzustandsdaten ersetzt werden, die von dem anderen Ausführungskern gesichert wurden.

20. Computersystem nach Anspruch 19, wobei das Initialisieren jedes Ausführungskerns die Schritte aufweist,

daß der Prozessor in den redundanten Ausführungsmodus umgeschaltet wird; und

daß Prozessorzustandsdaten aus den bestimmten Speicherplätzen in die diesen zugeordneten Ausführungskerne kopiert werden.

ZEICHNUNGEN SEITE 1

Nummer: DE 100 85 324 T 1
Int. Cl.[7]: G 06 F 11/16
Veröffentlichungstag: 5. Dezember 2002

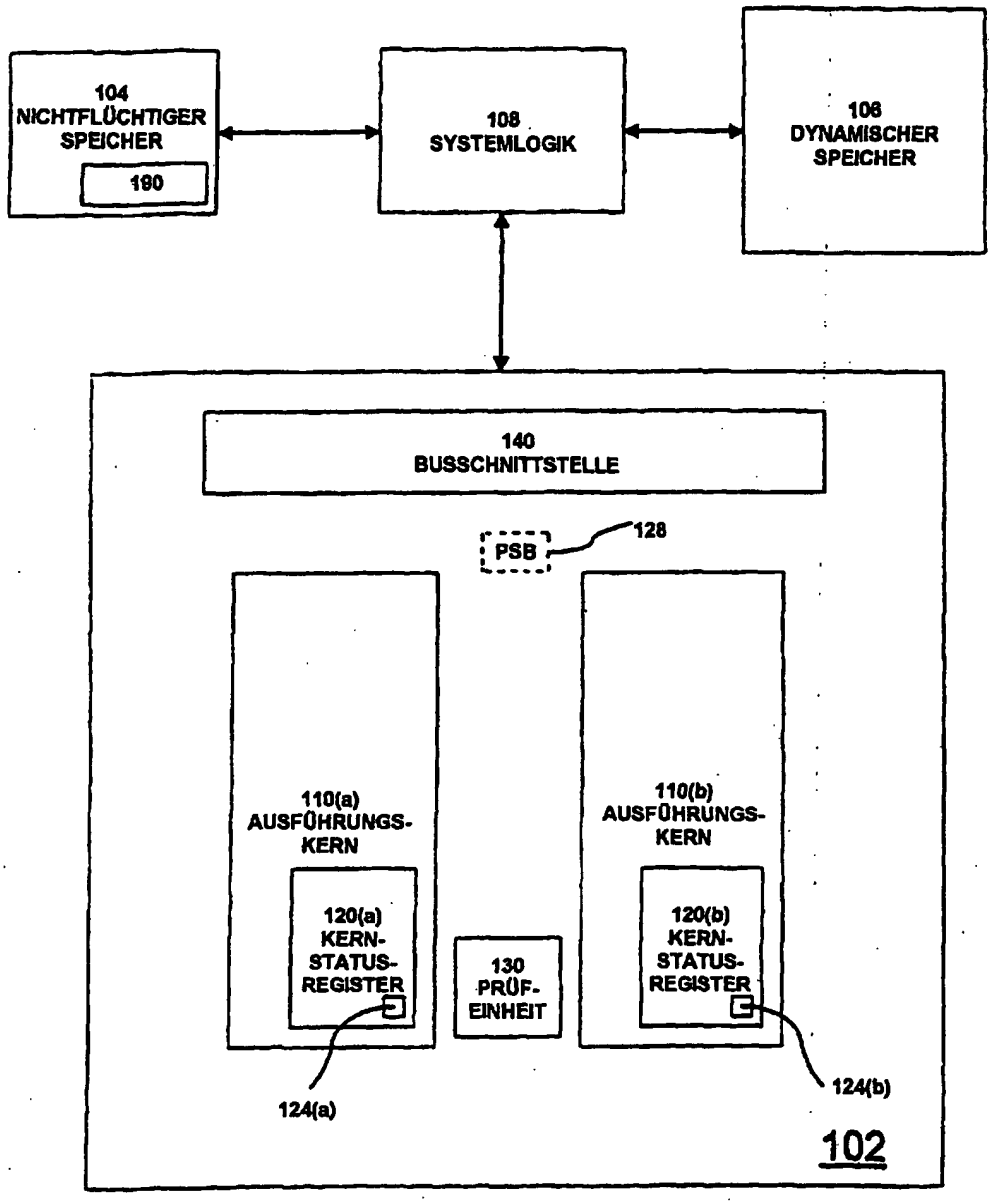

Fig. 1

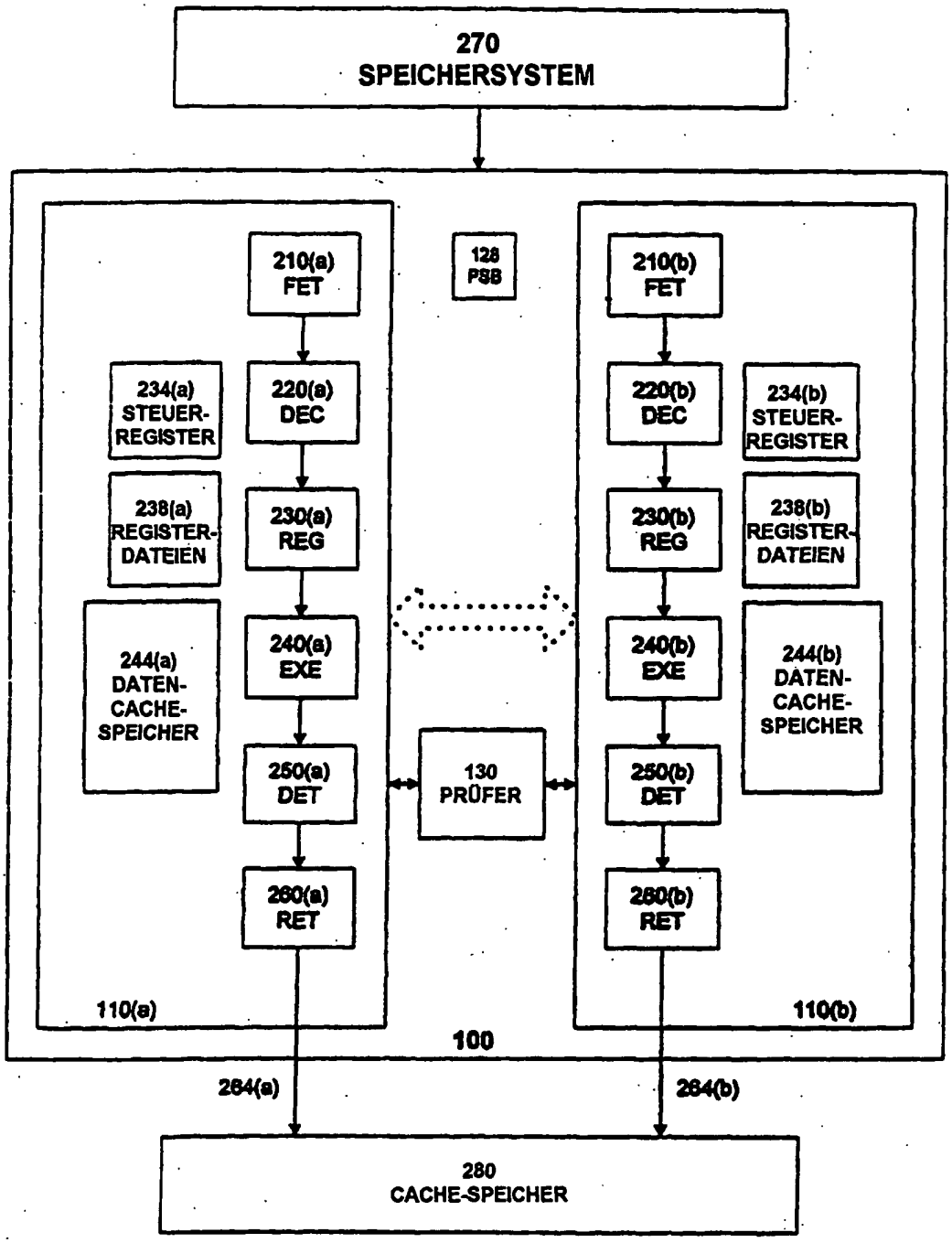

FIG. 2A

18.07.02

DE 100 85 324 T1

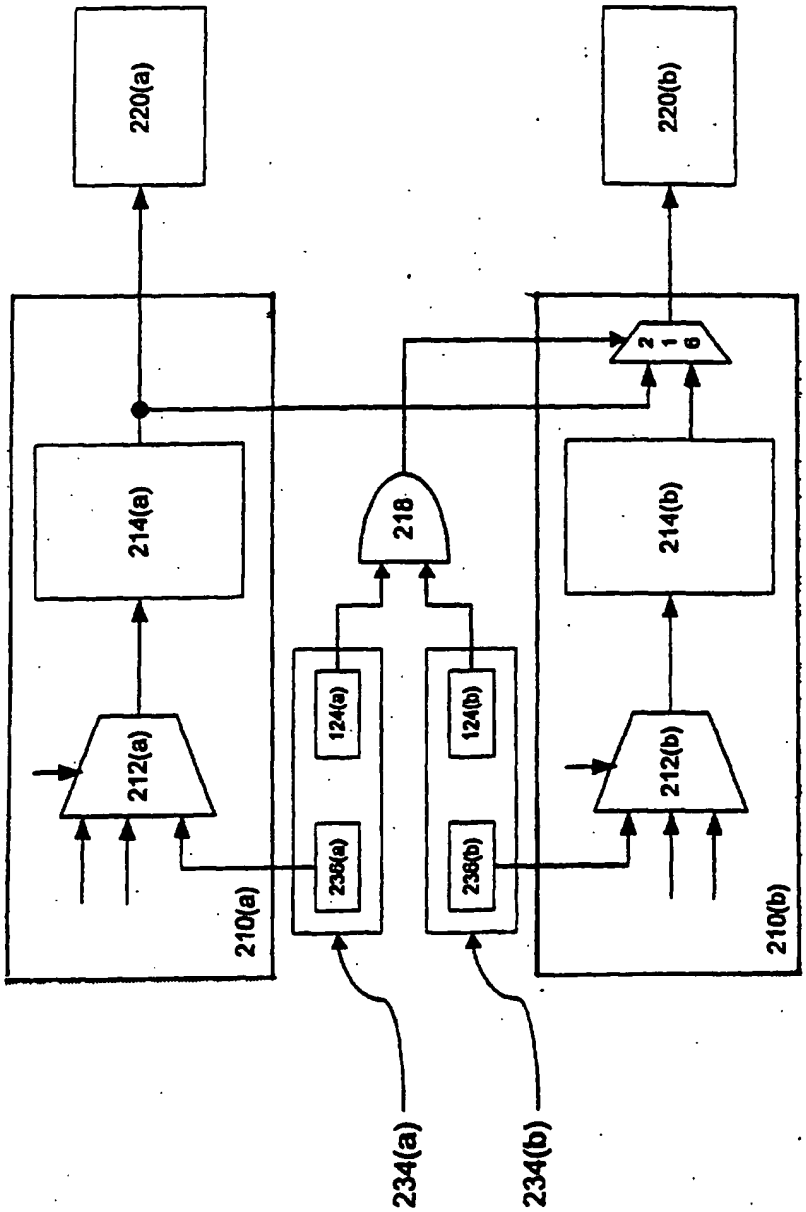

Fig. 2B

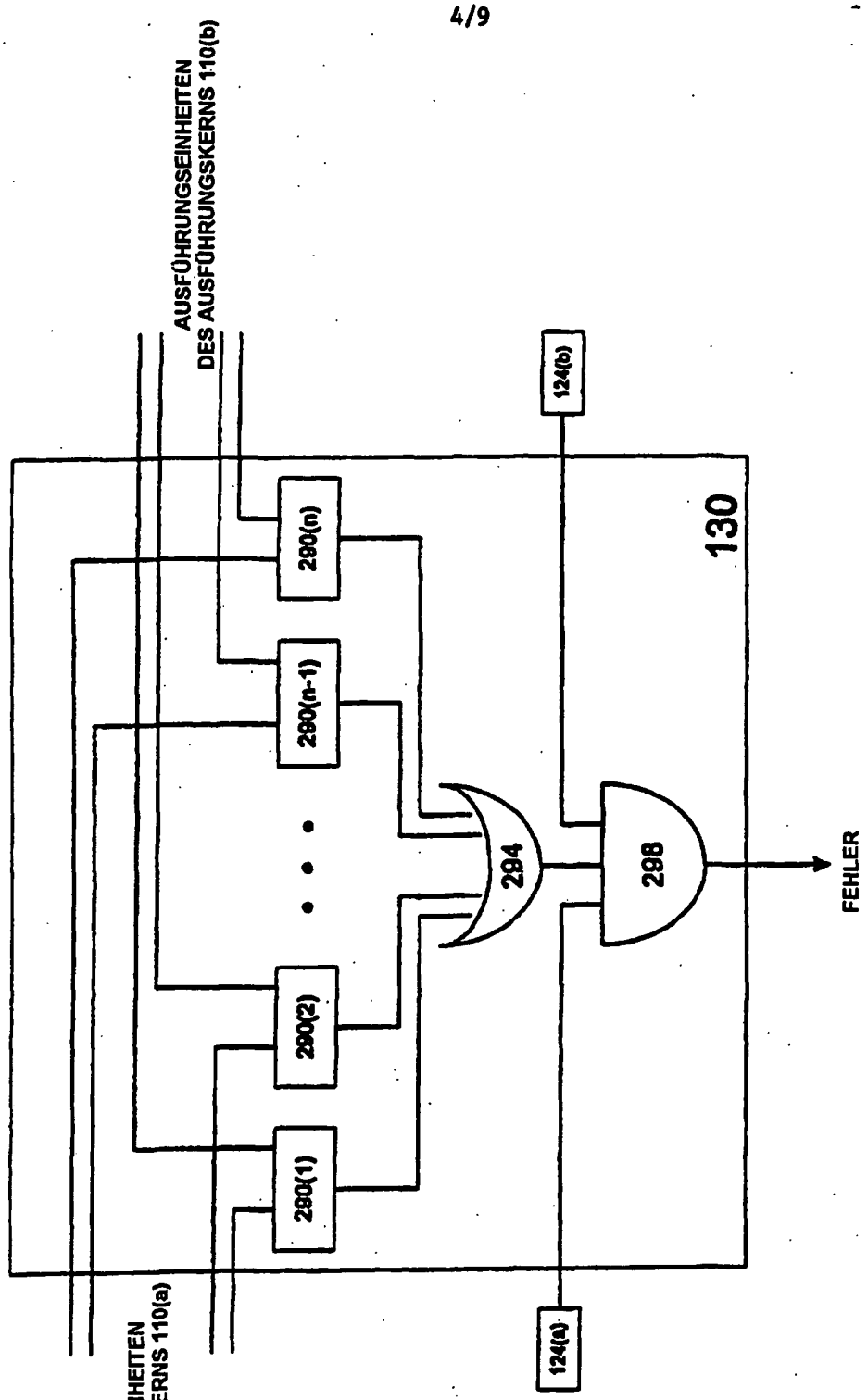
AUSFÜHRUNGSEINHEITEN DES AUSFÜHRUNGSKERNS 110(b)
AUSFÜHRUNGSEINHEITEN DES AUSFÜHRUNGSKERNS 110(a)
124(b)
124(a)
130
280(n)
280(n-1)
280(2)
280(1)
294
298
FEHLER

FIG. 2C

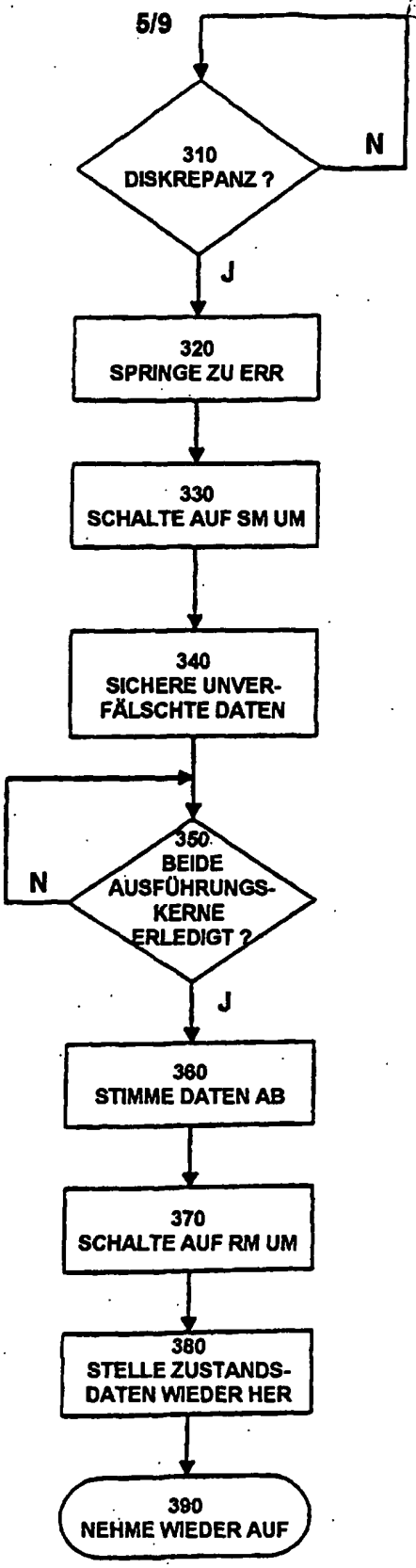

FIG. 3

DE 100 85 324 T1

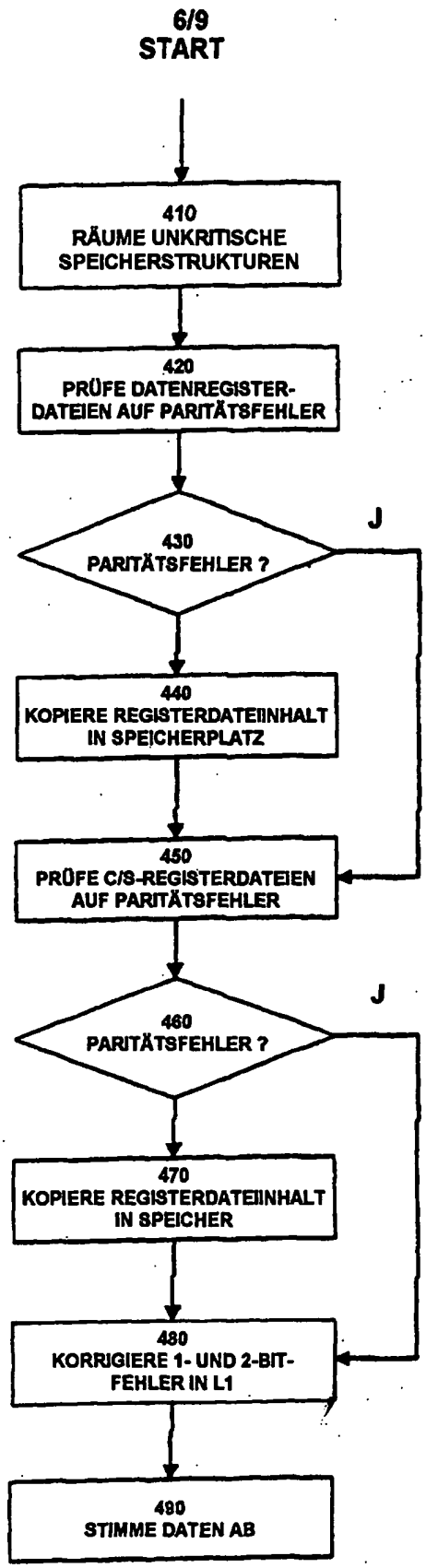

FIG. 4

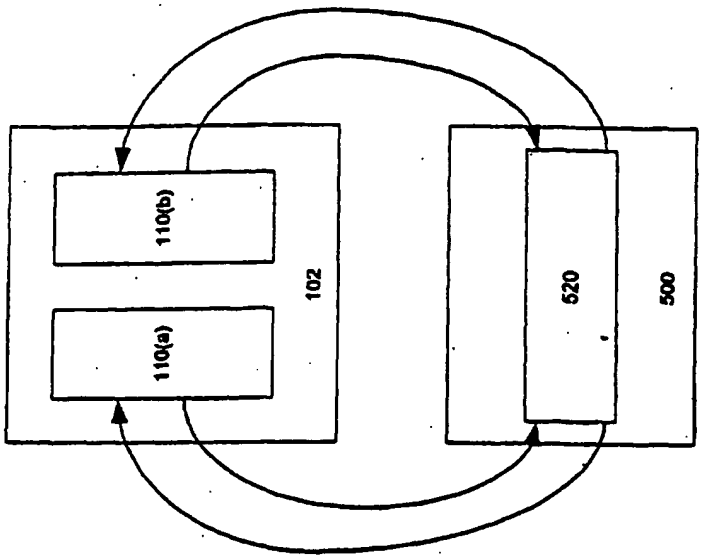

Fig. 5B

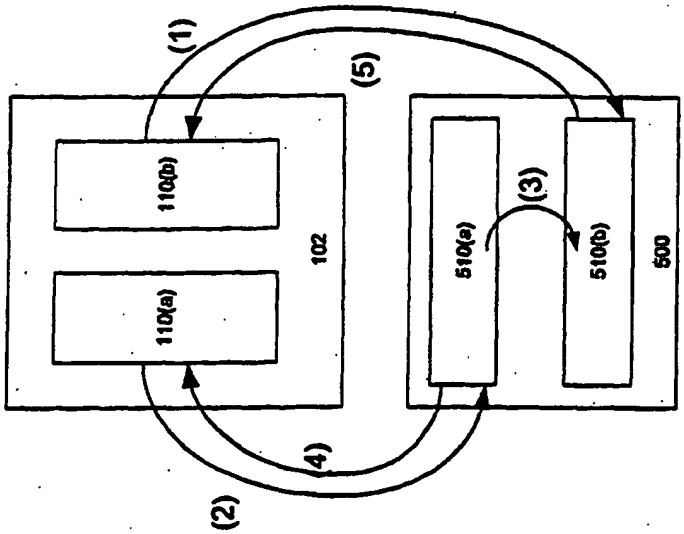

Fig. 5A

DE 100 85 324 T1

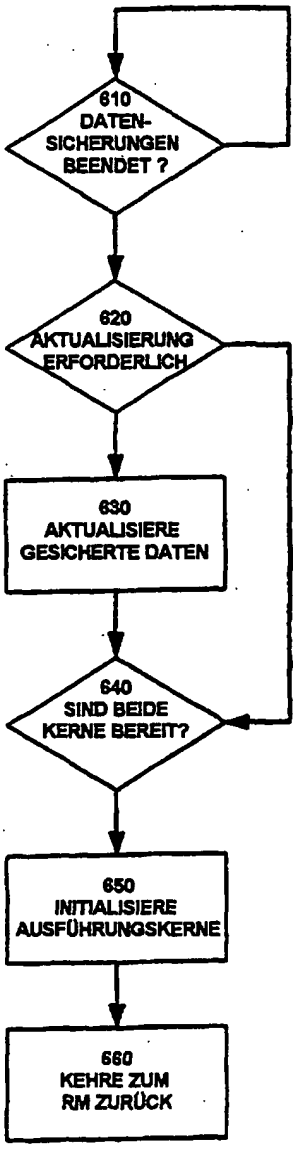

FIG. 6

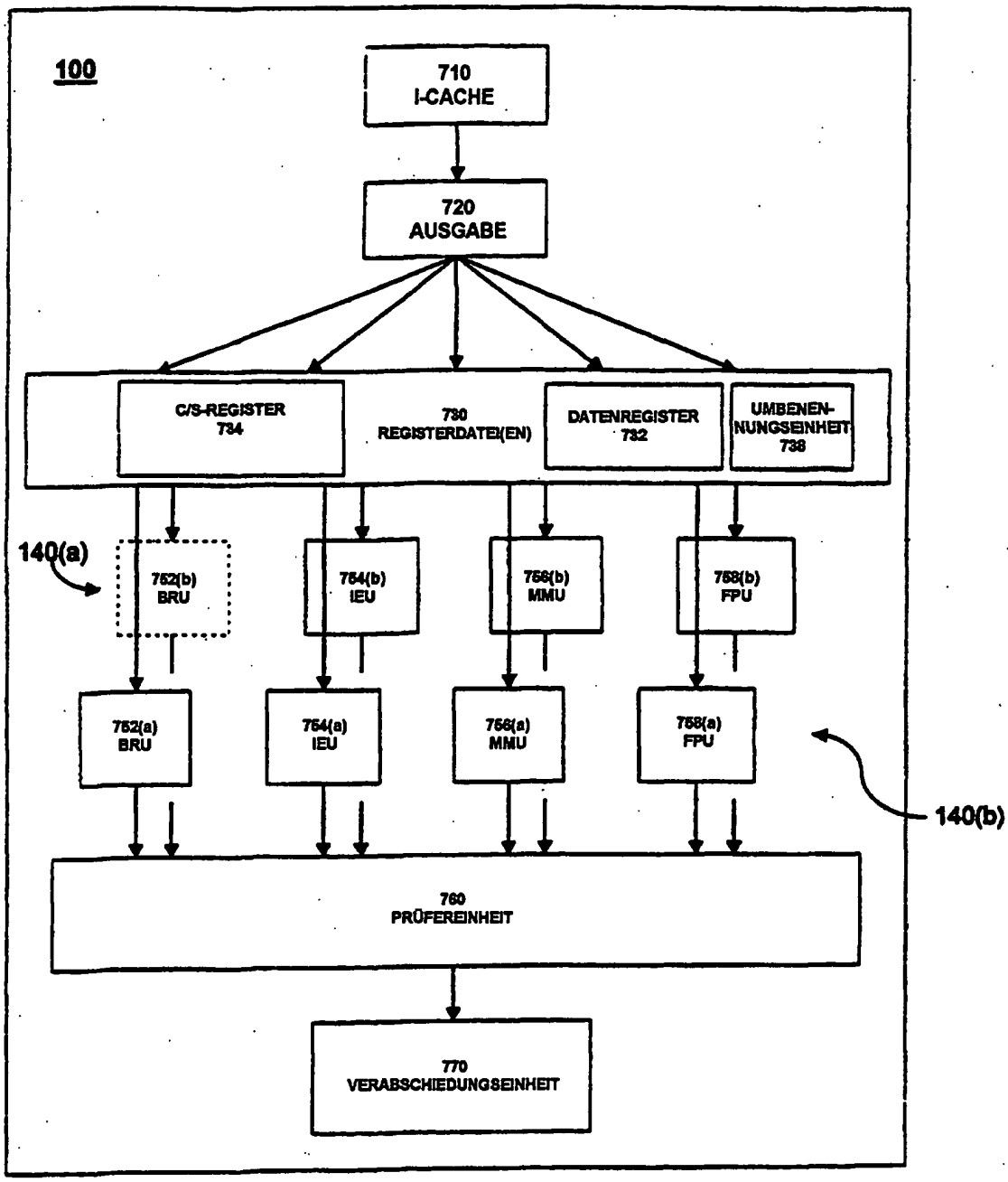

FIG. 7